The testhyphens package*

Claudio Beccari†

Abstract

This small file implements some code that was already published on TUGboat, but it is adapted to LaTeX and it is enriched with other commands. It helps those who create hyphenation patterns, as well normal LaTeX users, to test the correctness of the hyphenations of words lists. It works also with `xelatex` and `lualatex`.

Contents

1 Introduction

This small package introduces the declaration `\testhyphens` and the environment `checkhyphens`. The declaration is to be used within a group where the hyphenation parameters may be optionally set to any specific value; it processes a list of one or more space delimited words typesetting them one per line with hyphens between every syllable.

The language, the hyphenation rules of which are to be tested, may be the default language before entering the group, or can be a language specified with `\selectlanguage` within the group before activating the declaration: this declaration could be set within a `\foreignlangeuage{`⟨*language*⟩`}` argument or within the body of an `otherlanguage` environment.

The environment `checkhyphens` accepts an optional argument to set the hyphenation parameters, and a body consisting of a list of one or more space separated words; if the user just pastes a text copied from somewhere that included punctuation, the punctuation signs are ignored, but are not taken off from the text.

The main code was published long ago by Victor Eijkout, in "The bag of tricks", *TUGboat* 14.4 (1993), p. 424. He himself states that that code was created by

*This file has version number v.0.6, last revised on 2014/09/13.

†claudio dot beccari at gmail dot com

Jonathan Kew; Victor Eijkout just added an empty `\discretionary{}{}{}`; the idea came to him from the code created by Oliver Schoett that is used for the hyphenation exception list of TUGboat. Credit for the code goes completely to the authors mentioned above; I just added some sugar to use that code in LaTeX documents for the benefit of hyphenation pattern creators, and of regular users who want to know how a certain word or the words of a certain sentence would be hyphenated by TeX and friends.

Of course this package does not tell the whole truth: in fact TeX and friends hyphenate words only when certain conditions are met; these conditions are stated in Appendix H of the TeXbook, which is and remains the primary reference for understanding the hyphenation process used by TeX and friends.

Sometimes the user complains that certain words are not hyphenated; the user might use this package and find out that her/his words are regularly hyphenated; then what's happening? Simply those conditions are not met. Therefore the full understanding of the above mentioned Appendix H is fundamental before complaining about TeX not working properly with hyphenations and possibly before raising a bug notice.

2 Incompatibility with the previous version 0.5a

I had to change some definitions within the body of this package in order to avoid clashes mainly with the `french` language option of babel; in facts that option declares most punctuation marls as active characters. Unfortunately those characters receive global definitions, therefore they remain active even when they should not perform any action.

It is necessary to emphasize that the previous version used the colon as a parameter separator; while, in order to avoid conflicts with `french`, the parameter separator is a simple hyphen.

Defining active characters when using `polyglossia` may produce other incompatibilities that are easily circumvented by specifying languages in a different order. In facts, if you specify languages as such

```
\setmainlanguage[babelshorthands]{italian}
\setotherlanguages{english,french}
```

produces an error when `french` sets up its parameters for the UNICODE apostrophe: the cause of this error derives from the use of the `italian` option `babelshorthands`; of course if this option is not specified, no problems arise. But if you interchange the declarations as in

```
\setotherlanguages{english,french}
\setmainlanguage[babelshorthands]{italian}
```

no problems arise.

3 Usage

This package has been tested with `pdflatex`, `xelatex`, and `lualatex`; for `lualatex` a package `showhyphens` exists that marks the hyphen points of an entire document with thin red vertical strokes at the possible break points. The user may use the package s/he prefers by taking into consideration Her/his needs with respect with the features and the functionalities of package `testhyphens` compared to those of `showhyphens`.

Remember that normal Plain TEX and LATEX have available the command `\showhyphens` but, differently from this package, such command works only when the typesetting engine is `tex` or `pdftex`; moreover the result appears on the console window and in the `.log` file and no written record remains available to the user; for `xelatex` there was a workaround provided by another package, but for some reasons it did not work in every circumstance. This package `testhyphens`, on the opposite, prints the result in the output document and works also with `xetex` as the typesetting engine.

This simple package is loaded just with the usual statement

```
\usepackage{testhyphens}
```

The package defines for the end user one declaration `\testhyphens` and one environment `checkhyphens` the syntax of which is the following:

> `\begin{checkhyphens}[{`⟨*hyphenation parameters*⟩`}]`
> `{`⟨*word list*⟩`}`
> `\end{checklist}`

where ⟨*hyphenation parameters*⟩ is a *hyphen separated list* of two digits that in order will be assigned by the environment respectively to `\lefthyphenmin` and `\righthyphenmin`; ⟨*word list*⟩ is self explanatory. Nevertheless it good to remember that the environment isolates single words by using a space as a word separator; this requires that the list is preceded and followed by spaces; setting the environment with the opening and closing statements on single lines, as shown above, is sufficient, while a source line such as `\begin{checkhyphens}[{`⟨*hyphenation parameters*⟩`}]{`⟨*word list*⟩`}\end{checklist}` might not produce the desired result.

4 Using hyphenation parameters

The hyphenation parameters are just `\lefthyphenmin` and `\righthyphenmin`; they represent the minimum number of characters that the first and, respectively, the last word fragment must have before or until hyphenation takes place; for example in English the default values are 2 and 3; in Italian the default values are 2 and 2; in Greek the default values are 1 and 1.

Let us make an example: the word *idea* is spelt the same in English and Italian, although it is pronounced differently; but in both cases the word is not hyphenated at all in both languages. According to the Italian hyphenation rules its grammatical hyphenation is *i-de-a*; possibly in English the grammatical hyphenation is the

same; in any case the initial and the terminal syllables are too short to comply with the above mentioned TEX hyphenation parameters, and TEX refrains from hyphenating this word. In Greek, on the opposite, ιδέα is divided into ι-δέ-α.

If you use `checkhyphens` with the optional hyphenation parameters specified to 1 and 1 in Italian or in English, as in

```
checkhyphens[1:1] idea
```

you don't get any useful result in English, and the word remains un-hyphenated; but in Italian is turns out to be *i-de-a* as in Greek.

This happens because the Italian patterns were created by hand and tested with parameters 1 and 1, even if for typesetting purposes the parameters are set to 2 and 2. In English the patterns were created by means of the program `patgen` where parameters 2 and 3 were set. Typographically speaking the values 1 and 1 for Italian are almost useless, but in certain difficult narrow-measure texts the user has the possibility to locally set the first and/or the second parameter to 1, and solve that particular instance of problematic narrow-measure typesetting.

5 Examples

Here we show some actual examples using either the declaration or the environment. While typesetting in English we might verify the hyphenation of some words. We set in the source file the following lines (notice the `\par` command is a reminder that the group containing the declaration must be used in vertical mode; a blank line would be equivalent):

```
\par
{\testhyphens
manifests instruments
he analyses the samples and submits the analyses to the federal office
}
```

The result produced by this package is:
man-i-fests
in-stru-ments
he
anal-y-ses
the
sam-ples
and
sub-mits
the
anal-y-ses
to
the
fed-eral
of-fice

Notice the two instances of the word `analyses`; TEX hyphenates them the same way in English, but one of the two is hyphenated in a wrong way; unfortunately there is no way for TEX to distinguish homographic words that are not homophonic ones. In plain terms the verb "analyses" is stressed on the 'y', while the noun "analyses" is stressed on the second 'a'; the standard phonetic hyphenation rules for English probably should hyphenate the verb as "an-a-lyses", but TEX works only on the spelling and not on the sound and does not make any logical analysis of the text to find out if a word plays the rôle of a noun or of a verb. In this and similar situations (for example: "the record" and "to record") the user should check a good reliable dictionary and possibly use an explicit discretionary hyphen `\-`.

While typesetting in Italian it is possible to insert in the source file a set of lines such as these:

```
\begin{checkhyphens}
ebbe la bell'idea di viaggiare in scooter
\end{checkhyphens}
```

that produces the following result:
eb-be
la
bel-l'i-dea
di
viag-gia-re
in
scoo-ter

Notice that "idea" cannot be hyphenated by TEX when it is an isolated word; but in Italian the apostrophe is used (also) to mark a vocalic elision, therefore for hyphenation purposes, even if it is not a letter, it is assigned a lowercase code, and it legally becomes part of a word, in our case the word "bell'idea". Therefore the patterns that involve the apostrophe produce the result shown above, where the first grammatical hyphen shows up again.

While typesetting in Greek we might insert in the source file the following code:

```
\begin{checkhyphens}
επεξεργάζεται αρχεία τα οποία περιέχουν τουσ χαρακτήρες που
υπάρχουν στο πληκτρολόγιο του υπολογιστή σας
\end{checkhyphens}
```

and we would get:
ε-πε-ξερ-γά-ζε-ται
αρ-χεί-α
τα
ο-πο-ία
πε-ρι-έχουν
τουσ
χα-ρα-κτή-ρες
που

υ-πάρ-χουν
στο
πλη-κτρο-λόγιο
του
υ-πο-λο-γι-στή
σας

while, had we set the hyphenation parameters different from the default values 1 and 1, as in:

```
\begin{checkhyphens}[2-2]
επεξεργάζεται αρχεία τα οποία περιέχουν τουσ χαρακτήρες που
υπάρχουν στο πληκτρολόγιο του υπολογιστή σας
\end{checkhyphens}
```

the result would be
επε-ξερ-γά-ζε-ται
αρ-χεία
τα
οπο-ία
πε-ρι-έχουν
τουσ
χα-ρα-κτή-ρες
που
υπάρ-χουν
στο
πλη-κτρο-λόγιο
του
υπο-λο-γι-στή
σας

Notice that as things are at the moment of writing this documentations, the monotonic Greek hyphenation patterns fail to hyphenate after an accented vowel followed by a consonant; the grammar allows to hyphenate in that position; but it correctly recognises the hiathus between an unaccented vowel and an accented one; at the same time the first Greek example shows very clearly the effect of `\lefthyphenmin=1` compared to the second example where `\lefthyphenmin=2`. On the other hand, missing some hyphen points is certainly a feature that precludes full hyphenation, but it does not produce hyphenation errors.

For the user information it was the very use of this package that highlighted the features/bugs in the Greek hyphenation patterns; at the moment of writing this documentation, work is in progress for solving this behaviour. Notice that this behaviour is present when typesetting with `babel`, while these features are absent when `polyglossia` is used.

6 The code

The following code has very few modifications compared to the original code published by Victor Eijkout. For my own benefit I changed the declaration name from `\printhyphens` to `\t@sthyphens`; with an at-sign internal name it is less likely that the declaration is redefined by a user. I also moved the commend `\offinterlineskip` from its penultimate position in the original code to the first command to be executed within the `\vbox`; in this way I avoided certain cases in which the lack of interline skip would remain active after closing the environment.

The code is based on treating the interword space as an active character that plays different rôles according to its position; it lets the declaration isolate the words to be hyphenated and open suitable boxes, that are closed at the moment of processing the following word; the `\everypar` token-list distinguishes the various actions to be performed at each step.

But the trick is to set each word within a vertical box where the measure (the text width `\hsize`) is zero; in this way each word is hyphenated at the first hyphen point, but the word fragment that remains in the following line is still too long and gets hyphenated again; the process is repeated again and again until the initial string is exhausted. The contents of the `\vbox`, that now contains the syllables one per line, is reassembled to form one line, and this line is output within an `\hbox` and is therefore the box typeset in vertical mode; this is why the declaration must be issued in vertical mode, so as to start processing the word list with the right foot; this is also why the word list must start and end with spaces; the first one before the first word opens the first box, and the last space after the last word closes and outputs the last box.

```
\def\t@sthyphens{\everypar{\setbox0\lastbox \setbox1\hbox{\strut}\vbox\bgroup
\offinterlineskip
\everypar{\setbox0\lastbox \nobreak\hskip\z@}\dimen0=\hsize
\hsize=\z@ \hfuzz\maxdimen \def\par{\endgraf \hsize=\dimen0\getlastline
\egroup\endgraf}}\breakafterword}

\def\breakafterword{\catcode`\^^M\active\catcode`\ \active}

{\breakafterword\gdef^^M{\par}\global\let ^^M}

\def\getlastline{\setbox0\lastbox\ifvoid0\let\next\nomorelines
\else\unskip\unpenalty\setbox1\hbox{\unhbox0\strut\discretionary{}{}{}%
\unhbox1}\let\next\getlastline\fi\next}

\def\nomorelines{\unhbox1}
```

The above code is substantially the original one; now we define the user declaration control sequence `\testhyphens` by letting it to assume the meaning of the internal control sequence. Then we define the `checkhyphens` environment where we make sure that the whole process of the word list contained in its body takes place in vertical mode. In oder to process the optional argument for setting the hyphenation parameters, we define a delimited argument macro `\s@thyphenpars`

that uses the *hyphen* as argument delimiter.

```
\let\testhyphens\t@sthyphens

\newenvironment{checkhyphens}[1][\lefthyphenmin-\righthyphenmin]{%
\@tempcnta=\lefthyphenmin
\@tempcntb=\righthyphenmin
\s@thyphenpars[#1]\par\bgroup\t@sthyphens
}{%
\egroup\par
}

\def\s@thyphenpars[#1-#2]{%
\@tempcnta=#1\relax
\@tempcntb=#2\relax
\unless\ifnum\@tempcnta=\lefthyphenmin \lefthyphenmin=\@tempcnta\fi
\unless\ifnum\@tempcntb=\righthyphenmin \righthyphenmin=\@tempcntb\fi
}
```

That's all. Happy TEXing.