mfirstuc.sty v2.02: uppercasing first letter

Nicola L.C. Talbot

Dickimaw Books
http://www.dickimaw-books.com/

2015-12-17

Contents

1 Introduction

The mfirstuc package was originally part of the glossaries bundle for use with commands like `\Gls`, but as the commands provided by mfirstuc may be used without glossaries, the two have been split into separately maintained packages.

2 Capitalising a Word

A simple word can be capitalised just using the standard LaTeX upper casing command. For example,

```
\MakeUppercase word
```

but for commands like `\Gls` the word may be embedded within the argument of another command, such as a font changing command. This makes things more complicated for a general purpose solution, so the mfirstuc package provides:

\makefirstuc `\makefirstuc{⟨stuff⟩}`

This makes the first object of ⟨*stuff*⟩ upper case unless ⟨*stuff*⟩ starts with a control sequence followed by a non-empty group, in which case the first object in the group is converted to upper case. **No expansion is performed on the argument.**

If ⟨*stuff*⟩ starts with a control sequence that takes more than one argument, the case-changing will always be applied to the *first* argument. If the text that requires the case change is in one of the other arguments, you must hide the earlier arguments in a wrapper command. For example, instead of `\color{red}{text}` you need to define, say, `\red` as `\color{red}` and use `\red{text}`.

Examples:

- `\makefirstuc{abc}` produces Abc.
- `\makefirstuc{\emph{abc}}` produces *Abc* (`\MakeUppercase` has been applied to the letter "a" rather than `\emph`). Note however that

  ```
  \makefirstuc{{\em abc}}
  ```

  produces *ABC* (first object is `{\em abc}` so this is equivalent to `\MakeUppercase{\em abc}`), and

  ```
  {\makefirstuc{\em abc}}
  ```

  produces abc (`\em` doesn't have an argument therefore first object is `\em` and so is equivalent to `{\MakeUppercase{\em}abc}`).
- `\makefirstuc{{\'a}bc}` produces Ábc.
- `\makefirstuc{\ae bc}` produces Æbc.
- `\makefirstuc{{\ae}bc}` produces Æbc.
- `\makefirstuc{{ä}bc}` produces Äbc.

Note that non-Latin or accented characters appearing at the start of the text must be placed in a group (even if you are using the inputenc package). The reason for this restriction is detailed in Section 4.

In version 1.02 of mfirstuc, a bug fix resulted in a change in output if the first object is a control sequence followed by an empty group. Prior to version 1.02, `\makefirstuc{\ae{}bc}` produced æBc. However as from version 1.02, it now produces Æbc.

Note also that

```
\newcommand{\abc}{abc}
\makefirstuc{\abc}
```

produces: ABC. This is because the first object in the argument of `\makefirstuc` is `\abc`, so it does `\MakeUppercase{\abc}`. Whereas:

```
\newcommand{\abc}{abc}
\expandafter\makefirstuc\expandafter{\abc}
```

produces: Abc. There is a short cut command which will do this:

`\xmakefirstuc` `\xmakefirstuc{⟨stuff⟩}`

This is equivalent to `\expandafter\makefirstuc\expandafter{⟨stuff⟩}`. So

```
\newcommand{\abc}{abc}
\xmakefirstuc{\abc}
```

produces: Abc.

`\xmakefirstuc` only performs one level expansion on the *first* object in its argument. It does not fully expand the entire argument.

As from version 1.10, there is now a command that fully expands the entire argument before applying `\makefirstuc`:

`\emakefirstuc` `\emakefirstuc{⟨text⟩}`

Examples:

```
\newcommand{\abc}{\xyz a}
\newcommand{\xyz}{xyz}
No expansion: \makefirstuc{\abc}.
First object one-level expansion: \xmakefirstuc{\abc}.
Fully expanded: \emakefirstuc{\abc}.
```

produces: No expansion: XYZA. First object one-level expansion: XYZa. Fully expanded: Xyza.

If you use mfirstuc without the glossaries package, the standard `\MakeUppercase` command is used. If used with glossaries, `\MakeTextUppercase` (defined by the textcase package) is used instead. If you are using mfirstuc without the glossaries package and want to use `\MakeTextUppercase` instead, you can redefine

`\glsmakefirstuc` `\glsmakefirstuc{⟨text⟩}`

For example:

```
\renewcommand{\glsmakefirstuc}[1]{\MakeTextUppercase #1}
```

Remember to also load textcase (glossaries loads this automatically).

3 Capitalise Each Word in a Phrase or Sentence (Title Case)

New to mfirstuc v1.06:

\capitalisewords

\capitalisewords{⟨*text*⟩}

This command applies \makefirstuc to each word in ⟨*text*⟩ where the space character is used as the word separator. Note that it has to be a plain space character, not another form of space, such as ~ or \space. Note that no expansion is performed on ⟨*text*⟩.

Formatting for the entire phrase must go outside \capitalisewords (unlike \makefirstuc). Compare:

```
\capitalisewords{\textbf{a sample phrase}}
```

which produces:

A sample phrase

with:

```
\textbf{\capitalisewords{a sample phrase}}
```

which produces:

A Sample Phrase

\xcapitalisewords

\xcapitalisewords{⟨*text*⟩}

This is a short cut for \expandafter\capitalisewords\expandafter{⟨*text*⟩}.

As from version 1.10, there is now a command that fully expands the entire argument before applying \capitalisewords:

\ecapitalisewords

\ecapitalisewords{⟨*text*⟩}

Examples:

```
\newcommand{\abc}{\xyz\space four five}
\newcommand{\xyz}{one two three}
No expansion: \capitalisewords{\abc}.
First object one-level expansion: \xcapitalisewords{\abc}.
Fully expanded: \ecapitalisewords{\abc}.
```

produces: No expansion: ONE TWO THREE FOUR FIVE. First object one-level expansion: ONE TWO THREE four Five. Fully expanded: One Two Three Four Five.

(Remember that the spaces need to be explicit. In the second case above, using `\xcapitalisewords`, the space before "four" has been hidden within `\space` so it's not recognised as a word boundary, but in the third case, `\space` has been expanded to an actual space character.)

If you are using hyperref and want to use `\capitalisewords` or `\makefirstuc` (or the expanded variants) in a section heading, the PDF bookmarks won't be able to use the command as it's not expandable, so you will get a warning that looks like:

```
Package hyperref Warning: Token not allowed in a PDF string
(PDFDocEncoding):
(hyperref)                removing `\capitalisewords'
```

If you want to provide an alternative for the PDF bookmark, you can use hyperref's `\texorpdfstring` command. See the hyperref manual for further details.

Examples:

1. `\capitalisewords{a book of rhyme.}`

   produces: A Book Of Rhyme.

2. `\capitalisewords{a book\space of rhyme.}`

   produces: A Book of Rhyme.

3. ```
   \newcommand{\mytitle}{a book\space of rhyme.}
   \capitalisewords{\mytitle}
   ```

   produces: A BOOK OF RHYME. (No expansion is performed on `\mytitle`.) Compare with next example:

4. ```
   \newcommand{\mytitle}{a book\space of rhyme.}
   \xcapitalisewords{\mytitle}
   ```

   produces: A Book of Rhyme.

   However

   ```
   \ecapitalisewords{\mytitle}
   ```

   produces: A Book Of Rhyme.

As from v1.09, you can specify words which shouldn't be capitalised unless they occur at the start of ⟨*text*⟩ using:

`\MFUnocap` `\MFUnocap{`⟨*word*⟩`}`

This only has a local effect. The global version is:

\gMFUnocap

\gMFUnocap{⟨*word*⟩}

For example:

```
\capitalisewords{the wind in the willows}

\MFUnocap{in}%
\MFUnocap{the}%

\capitalisewords{the wind in the willows}
```

produces:

The Wind In The Willows
The Wind in the Willows

The list of words that shouldn't be capitalised can be cleared using

\MFUclear

\MFUclear

The package mfirstuc-english loads mfirstuc and uses \MFUnocap to add common English articles and conjunctions, such as "a", "an", "and", "but". You may want to add other words to this list, such as prepositions, but as there's some dispute over whether prepositions should be capitalised, I don't intend to add them to this package.

If you want to write a similar package for another language, all you need to do is create a file with the extension .sty that starts with

```
\NeedsTeXFormat{LaTeX2e}
```

The next line should identify the package. For example, if you have called the file mfirstuc-french.sty then you need:

```
\ProvidesPackage{mfirstuc-french}
```

It's a good idea to also add a version in the final optional argument, for example:

```
\ProvidesPackage{mfirstuc-french}[2014/07/30 v1.0]
```

Next load mfirstuc:

```
\RequirePackage{mfirstuc}
```

Now add all your \MFUnocap commands. For example:

```
\MFUnocap{de}
```

At the end of the file add:

```
\endinput
```

Put the file somewhere on TEX's path, and now you can use this package in your document. You might also consider uploading it to CTAN in case other users find it useful.

4 UTF-8

The \makefirstuc command works by utilizing the fact that, in most cases, TEX doesn't require a regular argument to be enclosed in braces if it only consists of a single token. (This is why you can do, say, \frac12 instead of \frac{1}{2} or x^2 instead of x^{2}, although some users frown on this practice.)

A simplistic version of the \makefirstuc command is:

```
\newcommand*{\FirstUC}[1]{\MakeUppercase #1}
```

Here

```
\FirstUC{abc}
```

is equivalent to

```
\MakeUppercase abc
```

and since \MakeUppercase requires an argument, it grabs the first token (the character "a" in this case) and uses that as the argument so that the result is: Abc.

The glossaries package needs to take into account the fact that the text may be contained in the argument of a formatting command, such as \acronymfont, so \makefirstuc has to be more complicated than the trivial \FirstUC shown above, but at its basic level, \makefirstuc uses this same method and is the reason why, in most cases, you don't need to enclose the first character in braces. So if

\MakeUppercase ⟨*stuff*⟩

works, then

\makefirstuc{⟨*stuff*⟩}

should also work and so should

\makefirstuc{\foo{⟨*stuff*⟩}}

but if

\MakeUppercase ⟨*stuff*⟩

doesn't work, then neither will

\makefirstuc{⟨*stuff*⟩}

nor

\makefirstuc{\foo{⟨*stuff*⟩}}

Try the following document:

```
\documentclass{article}

\usepackage[utf8]{inputenc}
\usepackage[T1]{fontenc}

\begin{document}

\MakeUppercase ãbc

\end{document}
```

This will result in the error:

```
! Argument of \UTFviii@two@octets has an extra }.
```

This is why `\makefirstuc{ãbc}` won't work. It will only work if the character ã is placed inside a group.

The reason for this error message is due to TEX having been written before Unicode was invented. Although ã may look like a single character in your text editor, from TEX's point of view it's *two* tokens. So

```
\MakeUppercase ãbc
```

tries to apply `\MakeUppercase` to just the first octet of ã. This means that the second octet has been separated from the first octet, which is the cause of the error. In this case the argument isn't a single token, so the two tokens (the first and second octet of ã) must be grouped:

```
\MakeUppercase{ã}bc
```

Note that XƎTEX (and therefore XƎLATEX) is a modern implementation of TEX designed to work with Unicode and therefore doesn't suffer from this drawback. Now let's look at the XƎLATEX equivalent of the above example:

```
\documentclass{article}

\usepackage{fontspec}

\begin{document}

\MakeUppercase ãbc

\end{document}
```

This works correctly when compiled with XƎLATEX. This means that `\makefirstuc{ãbc}` will work *provided you use XƎLATEX and the fontspec package.*