mfirstuc.sty v2.07: uppercasing first letter

Nicola L.C. Talbot

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

2021-10-15

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.

The commands described here all have limitations. To minimise problems, use text-block style semantic commands with one argument (the text that requires case-changing), and avoid scoped declarations.

Here are some examples of semantic commands:

1. Quoted material:

   ```
   \newcommand{\qt}[1]{``#1''}
   ```

   (or use the csquotes package). With this, the following works:

   ```
   \makefirstuc{\qt{word}}
   ```

   This produces:

   “Word”

   Whereas

   ```
   \makefirstuc{``word''}
   ```

   fails (no case-change and double open quote becomes two single open quotes):

   ‘‘word”

2. Font styles or colours:

   ```
   \newcommand*{\alert}[1]{\textcolor{red}{#1}}
   ```

   Now the following is possible:

   ```
   \makefirstuc{\alert{word}}
   ```

This produces

Word

Note that `\makefirstuc{\textcolor{red}{word}}` fails (with an error) because the case-changing interferes with the label.

Define these semantic commands robustly if you intend using any of the commands that fully expand their argument (`\emakefirstuc`, `\ecapitalisewords` and `\ecapitalisefmtwords`).

2 Capitalising the First Letter of 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 punctuation that should be skipped over, use:

`\MFUskippunc` `\MFUskippunc{⟨punctuation⟩}`

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 `\textcolor{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.
- `\makefirstuc{\MFUskippunc{'}tis the season}` produces 'Tis the season.

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

New to version 2.04: There is now limited support for UTF-8 characters with the inputenc package, provided that you load datatool-base (at least v2.24) before mfirstuc (datatool-base is loaded automatically with newer versions of glossaries). If available mfirstuc will now use datatool-base's `\dtl@getfirst@UTFviii` command which is still experimental. See the datatool manual for further details.

```
\documentclass{article}

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

\usepackage{datatool-base}[2016/01/12]% v2.24+
\usepackage{mfirstuc}

\begin{document}
\makefirstuc{élite}
\end{document}
```

(Package ordering is important.)

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 the First Letter of 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*⟩. See Section 3.2 for excluding words (such as "of") from the case-changing.

The actual capitalisation of each word is done using

\MFUcapwordfirstuc

```
\MFUcapwordfirstuc{⟨word⟩}
```

This just does \makefirstuc{⟨*word*⟩} by default. There's a conditional that determines whether or not to consider a hyphen a word break:

\ifMFUhyphen

```
\ifMFUhyphen
```

If you want to title case each part of a compound word containing hyphens, you can enable this using

\MFUhyphentrue

```
\MFUhyphentrue
```

or switch it back off again using:

\MFUhyphenfalse

```
\MFUhyphenfalse
```

Compare

```
\capitalisewords{server-side includes}
```

which produces:

Server-side Includes

with

```
\MFUhyphentrue
\capitalisewords{server-side includes}
```

which produces:

Server-Side Includes

Note that this won't apply exceptions to each part of the hyphenated word.

For other punctuation you need to markup the character with:

\MFUwordbreak `\MFUwordbreak{⟨punctuation⟩}`

For example:

```
\capitalisewords{a big\MFUwordbreak{/}small idea}
```

which produces:

A Big/Small Idea

In this case, exceptions are applied. For example, if mfirstuc-english is loaded then:

```
\capitalisewords{one and\MFUwordbreak{/}or another}
```

produces:

One and/or Another

Note that you can't hide `\MFUwordbreak` inside a command or group.

Exceptions aren't applied if `\MFUwordbreak` occurs before the first space. For example:

```
\MFUnocap{a}\MFUnocap{the}%
\capitalisewords{a\MFUwordbreak{/}the something}
```

produces:

A/The Something

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

As from version 2.03, there is now a command for phrases that may include a formatting command:

```
\capitalisefmtwords{⟨phrase⟩}
```

where ⟨*phrase*⟩ may be just words (as with `\capitalisewords`) or may be entirely enclosed in a formatting command in the form

```
\capitalisefmtwords{⟨cs⟩{⟨phrase⟩}}
```

or contain formatted sub-phrases

```
\capitalisefmtwords{⟨words⟩ ⟨cs⟩{⟨sub-phrase⟩} ⟨words⟩}
```

Avoid scoped declarations.

`\capitalisefmtwords` is only designed for phrases that contain text-block commands with a single argument, which should be a word or sub-phrase. Anything more complicated is likely to break. Instead, use the starred form or `\capitalisewords`.

The starred form only permits a text-block command at the start of the phrase. Examples:

1. Phrase entirely enclosed in a formatting command:

   ```
   \capitalisefmtwords{\textbf{a small book of rhyme}}
   ```

   produces:

   A Small Book Of Rhyme

2. Sub-phrase enclosed in a formatting command:

   ```
   \capitalisefmtwords{a \textbf{small book} of rhyme}
   ```

produces:

A **Small Book** Of Rhyme

3. Nested text-block commands:

```
\capitalisefmtwords{\textbf{a \emph{small book}} of rhyme}
```

produces:

A *Small Book* Of Rhyme

4. Indicating words that shouldn't have the case changed (see Section 3.2):

```
\MFUnocap{of}
\capitalisefmtwords{\textbf{a \emph{small book}} of rhyme}
```

produces:

A *Small Book* of Rhyme

5. Starred form:

```
\MFUnocap{of}
\capitalisefmtwords*{\emph{a small book of rhyme}}
```

produces:

A Small Book of Rhyme

6. The starred form also works with just text (no text-block command):

```
\MFUnocap{of}
\capitalisefmtwords*{a small book of rhyme}
```

produces:

A Small Book of Rhyme

If there is a text-block command within the argument of the starred form, it's assumed to be at the start of the argument. Unexpected results can occur if there are other commands. For example

```
\MFUnocap{of}
\capitalisefmtwords*{\emph{a small} book \textbf{of rhyme}}
```

produces:

A Small Book **Of rhyme**

(In this case `\textbf{of rhyme}` is considered a single word.) Similarly if the text-block command occurs in the middle of the argument:

```
\MFUnocap{of}
\capitalisefmtwords*{a \emph{very small} book of rhyme}
```

produces:

A *Very small* Book of Rhyme

(In this case `\emph{very small}` is considered a single word.)

Grouping causes interference:

```
\capitalisefmtwords{{a \emph{small book}} of rhyme}
```

produces:

A *Small book* Of Rhyme

As with all the commands described here, avoid declarations. For example, the following fails:

```
\capitalisefmtwords{{\bfseries a \emph{small book}} of rhyme}
```

produces:

a *Small book* Of Rhyme

Avoid complicated commands in the unstarred version. For example, the following breaks:

```
\newcommand*{\swap}[2]{{#2}{#1}}
\capitalisefmtwords{a \swap{bo}{ok} of rhyme}
```

However it works okay with the starred form and the simpler `\capitalisewords`:

```
\newcommand*{\swap}[2]{{#2}{#1}}
\capitalisefmtwords*{a \swap{bo}{ok} of rhyme}

\capitalisewords{a \swap{bo}{ok} of rhyme}
```

Produces:

A okBo Of Rhyme
A okBo Of Rhyme

Note that the case change is applied to the first argument.

\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⟩}`

There are also similar shortcut commands for the version that allows text-block commands:

\xcapitalisefmtwords `\xcapitalisefmtwords{⟨text⟩}`

The unstarred version is a short cut for `\expandafter\capitalisefmtwords\expandafter{⟨text⟩}`. Similarly the starred version of `\xcapitalisefmtwords` uses the starred version of `\capitalisefmtwords`.

For full expansion:

\ecapitalisefmtwords `\ecapitalisefmtwords{⟨text⟩}`

Take care with this as it may expand non-robust semantic commands to replacement text that breaks the functioning of `\capitalisefmtwords`. Use robust semantic commands where possible. Again this has a starred version that uses the starred form of `\capitalisefmtwords`.

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.)

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. (`\space` has been expanded to an actual space character.)

5. ```
   \newcommand*{\swap}[2]{{#2}{#1}}
   \capitalisewords{a \swap{bo}{ok} of rhyme}

   \ecapitalisewords{a \swap{bo}{ok} of rhyme}
   ```

   produces:

   A okBo Of Rhyme

   A OKbo Of Rhyme

   This is because the argument of `\ecapitalisewords` is fully expanded before being passed to `\capitalisewords` so that last example is equivalent to:

   ```
   \capitalisewords{a {ok}{bo} of rhyme}
   ```
   ```

3.1 PDF Bookmarks

If you are using hyperref and want to use `\capitalisewords`, `\capitalisefmtwords` 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. For example:

```
\chapter{\texorpdfstring
  {\capitalisewords{a book of rhyme}}% TeX
  {A Book of Rhyme}% PDF
}
```

Alternatively, you can use hyperref's mechanism for disabling commands within the bookmarks. For example:

```
\pdfstringdefDisableCommands{%
 \let\capitalisewords\@firstofone
}
```

See the hyperref manual for further details.

3.2 Excluding Words From Case-Changing

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
```

You can also simply place an empty group in front of a word if you don't want that specific instance to be capitalised. For example:

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

produces:

The wind In The Willows

This is also a useful way of protecting commands that shouldn't be parsed. For example:

```
\capitalisewords{this is section {}\nameref{sec:nocap}.}
```

produces

This Is Section Excluding Words From Case-Changing.

(No case-changing is applied to `\nameref{sec:nocap}`. It just happens to already be in title case.)

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

(See Binary Files, Text Files and File Encodings if you are confused about how file encodings such as UTF-8 relate to text files.)

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
```

Over recent years the LATEX kernel has changed to allow the use of UTF-8 characters within labels but the fundamental problem of each octet been viewed as a separated token remains.

Note that XETEX (and therefore XELATEX) 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 XELATEX equivalent of the above example:

```
\documentclass{article}

\usepackage{fontspec}

\begin{document}
```

```
\MakeUppercase ãbc

\end{document}
```

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

Version 2.24 of datatool-base added the command `\dtl@getfirst@UTFviii` which attempts to grab both octets. If this command has been defined, mfirstuc will use it when it tries to split the first character from the rest of the word. See the datatool documented code for further details.

Index