mfirstuc.sty v1.09: uppercasing first letter

Nicola L.C. Talbot

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

2014-07-30

The glossaries bundle includes the package mfirstuc which provides the command:

\makefirstuc

\makefirstuc{⟨*stuff*⟩}

This makes the first object of ⟨*stuff*⟩ uppercase 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 uppercase. **No expansion is performed on the argument.**

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 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) due to expansion issues.

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.

If you use mfirstuc without the glossaries package, the standard `\MakeUppercase` command is used. If used with glossaries, `\MakeTextUppercase` (defined by textcase the 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).

New to mfirstuc v1.06:

\capitalisewords

\capitalisewords{⟨*text*⟩}

This command apply \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*⟩.

\xcapitalisewords

\xcapitalisewords{⟨*text*⟩}

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

If you are using hyperref and want to use \capitalisewords or \makefirstuc (or \xcapitalisewords/\xmakefirstuc) 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, so ⟨*text*⟩ consists of just one "word".) Compare with next example:

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

\fi `\MFUnocap{⟨word⟩}`

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

\fi `\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.