fmtcount.sty v1.04: Displaying the Values of LaTeX Counters

Nicola L.C. Talbot

29 July 2006

Contents

1 Introduction

The fmtcount package provides commands to display the values of LaTeX counters in a variety of formats. It also provides equivalent commands for actual numbers rather than counter names. Limited multilingual support is available.

2 Installation

This package is distributed with the files `fmtcount.dtx` and `fmtcount.ins`. To extract the code do:

```
latex fmtcount.ins
```

This will create the files `fmtcount.sty` and `fmtcount.perl`, along with several `.def` files. Place `fmtcount.sty` and the `.def` files somewhere where LaTeX will find them (e.g. `texmf/tex/latex/fmtcount/`) and place `fmtcount.perl` somewhere where LaTeX2HTML will find it (e.g. latex2html/styles). Remember to refresh the TeX database (using `texhash` under Linux, for other operating systems check the manual.)

3 Available Commands

The commands can be divided into two categories: those that take the name of a counter as the argument, and those that take a number as the argument.

`\ordinal` `\fmtord` The macro `\ordinal{`⟨*counter*⟩`}` will print the value of a LaTeX counter ⟨*counter*⟩ as an ordinal, where the macro `\fmtord{`⟨*text*⟩`}` is used to format the st,nd,rd,th bit. By default the ordinal is formatted as a superscript, if the package option `level` is used, it is level with the text. For example, if the current section is 3, then `\ordinal{section}` will produce the output: 3rd.

Note: the memoir class also defines a command called `\ordinal` which takes a number as an argument instead of a counter. In order to overcome this incompatiblity, if you want to use the fmtcount package with the memoir class you should use `\FCordinal` to access fmtcount's version of `\ordinal`, and use `\ordinal` to use memoir's version of that command.

`\ordinalnum` The macro `\ordinalnum` is like `\ordinal` but takes an actual number rather than a counter as the argument. For example: `\ordinalnum{3}` will produce: 3rd.

`\numberstring` `\Numberstring` The macro `\numberstring{`⟨*counter*⟩`}` will print the value of ⟨*counter*⟩ as text. E.g. `\numberstring{section}` will produce: three. The macro `\Numberstring{`⟨*counter*⟩`}` does the same as `\numberstring`, but with initial letters in uppercase. For example, `\Numberstring{section}` will produce: Three.

`\numberstringnum` `\Numberstringnum` The macros `\numberstringnum` and `\Numberstringnum` work like `\numberstring` and `\Numberstring`, respectively, but take an actual number rather than a counter as the argument. For example: `\Numberstringnum{105}` will produce: One Hundred and Five.

`\ordinalstring` `\Ordinalstring` The macro `\ordinalstring{`⟨*counter*⟩`}` will print the value of ⟨*counter*⟩ as a textual ordinal. E.g. `\ordinalstring{section}` will produce: third. The macro `\Ordinalstring{`⟨*counter*⟩`}` does the same as `\ordinalstring`, but with initial letters in uppercase. For example, `\Ordinalstring{section}` will produce: Third.

`\ordinalstringnum` `\Ordinalstringnum` The macros `\ordinalstringnum` and `\Ordinalstringnum` work like `\Ordinalstring` and `\Ordinalstring`, respectively, but take an actual number rather than a counter as the argument. For example, `\ordinalstringnum{3}` will produce: third.

`\binary` The macro `\binary{`⟨*counter*⟩`}` will print the value of ⟨*counter*⟩ as a binary number. E.g. `\binary{section}` will produce: 11. The declaration `\padzeroes[`⟨*n*⟩`]` will ensure numbers are written to ⟨*n*⟩ digits, padding with zeroes if necessary. E.g. `\padzeroes[8]\binary{section}` will produce: 00000011. The default value for ⟨*n*⟩ is 17.

`\padzeroes`

`\binarynum` The macro `\binarynum` is like `\binary` but takes an actual number rather than a counter as the argument. For example: `\binarynum{5}` will produce: 101.

`\octal` The macro `\octal{`⟨*counter*⟩`}` will print the value of ⟨*counter*⟩ as an octal number. For example, if you have a counter called, say `mycounter`, and you set the value to 125, then `\octal{mycounter}` will produce: 177. Again, the number will be padded with zeroes if necessary, depending on whether `\padzeroes` has been used.

`\octalnum` The macro `\octalnum` is like `\octal` but takes an actual number rather than a counter as the argument. For example: `\octalnum{125}` will produce: 177.

`\hexadecimal` The macro `\hexadecimal{`⟨*counter*⟩`}` will print the value of ⟨*counter*⟩ as a hexadecimal number. Going back to the previous example, `\hexadecimal{mycounter}` will produce: 7d. Again, the number will be padded with zeroes if necessary, de-

\Hexadecimal pending on whether \padzeroes has been used. \Hexadecimal{⟨*counter*⟩} does the same thing, but uses uppercase characters, e.g. \Hexadecimal{mycounter} will produce: 7D.

\hexadecimalnum \Hexadecimalnum The macros \hexadecimalnum and \Hexadecimalnum are like \hexadecimal and \Hexadecimal but take an actual number rather than a counter as the argument. For example: \hexadecimalnum{125} will produce: 7d, and \Hexadecimalnum{125} will produce: 7D.

\decimal The macro \decimal{⟨*counter*⟩} is similar to \arabic but the number can be padded with zeroes depending on whether \padzeroes has been used. For example: \padzeroes[8]\decimal{section} will produce: 00000005.

\decimalnum The macro \decimalnum is like \decimal but takes an actual number rather than a counter as the argument. For example: \padzeroes[8]\decimalnum{5} will produce: 00000005.

\aaalph The macro \aaalph{⟨*counter*⟩} will print the value of ⟨*counter*⟩ as: a b ... z aa bb ... zz etc. For example, \aaalpha{mycounter} will produce: uuuuu

\AAAlph if mycounter is set to 125. \AAAlph{⟨*counter*⟩} does the same thing, but uses uppercase characters, e.g. \AAAlph{mycounter} will produce: UUUUU.

\aaalphnum The macros \aaalphnum and \AAAlphnum are like \aaalph and \AAAlph but

\AAAlphnum take an actual number rather than a counter as the argument. For example: \aaalphnum{125} will produce: uuuuu, and \AAAlphnum{125} will produce: UUUUU.

\abalph The macro \abalph{⟨*counter*⟩} will print the value of ⟨*counter*⟩ as: a b ... z aa ab ... az etc. For example, \abalpha{mycounter} will produce: du

\ABAlph if mycounter is set to 125. \ABAlph{⟨*counter*⟩} does the same thing, but uses uppercase characters, e.g. \ABAlph{mycounter} will produce: DU.

\abalphnum The macros \abalphnum and \ABAlphnum are like \abalph and \ABAlph but

\ABAlphnum take an actual number rather than a counter as the argument. For example: \abalphnum{125} will produce: du, and \ABAlphnum{125} will produce: DU.

4 Package Options

The following options can be passed to this package:

raise make ordinal st,nd,rd,th appear as superscript

level make ordinal st,nd,rd,th appear level with rest of text

These can also be set using the command:

\fmtcountsetoptions \fmtcountsetoptions{fmtord=⟨*type*⟩}

where ⟨*type*⟩ is either level or raise.

5 Multilingual Support

Version 1.02 of the fmtcount package now has limited multilingual support. The following languages are implemented: English, Spanish, Portuguese, French, French (Swiss) and French (Belgian). The package checks to see if the command \date⟨*language*⟩ is defined[1], and will load the code for those languages. The commands \ordinal, \ordinalstring and \numberstring (and their variants) will then be formatted in the currently selected language.

[1] this will be true if you have loaded babel

If the French language is selected, the French (France) version will be used by default (e.g. soxiante-dix for 70). To select the Swiss or Belgian variants (e.g. septente for 70) use: `\fmtcountsetoptions{french=`⟨*dialect*⟩`}` where ⟨*dialect*⟩ is either `swiss` or `belgian`. You can also use this command to change the action of `\ordinal`. `\fmtcountsetoptions{abbrv=true}` to produce ordinals of the form 2^{e} or `\fmtcountsetoptions{abbrv=false}` to produce ordinals of the form 2^{eme} (default).

The `french` and `abbrv` settings only have an effect if the French language has been defined.

The male gender for all languages is used by default, however the feminine form can be obtained by passing `f` as an optional argument to `\ordinal`, `\ordinalnum` etc. For example: `\numberstring{section}[f]`. Note that the optional argument comes *after* the compulsory argument.

Let me know if you find any spelling mistakes (has been known to happen in English, let alone other languages I'm not so familiar with.)

6 Configuration File `fmtcount.cfg`

You can save your preferred default settings to a file called `fmtcount.cfg`, and place it on the TEX path. These settings will then be loaded by the fmtcount package.

Note that if you are using the datetime package, the `datetime.cfg` configuration file will override the `fmtcount.cfg` configuration file. For example, if `datetime.cfg` has the line:

```
\renewcommand{\fmtord}[1]{\textsuperscript{\underline{#1}}}
```

and if `fmtcount.cfg` has the line:

```
\fmtcountsetoptions{fmtord=level}
```

then the former definition of `\fmtord` will take precedence.

7 LaTeX2HTML style

The LATEX2HTML style file `fmtcount.perl` is provided. The following limitations apply:

- `\padzeroes` only has an effect in the preamble.
- The configuration file `fmtcount.cfg` is currently ignored. (This is because I can't work out the correct code to do this. If you know how to do this, please let me know.) You can however do:

  ```
  \usepackage{fmtcount}
  \html{\input{fmtcount.cfg}}
  ```

  This, I agree, is an unpleasant cludge.

8 Acknowledgements

I would like to thank my mother for the French and Portuguese support and my Spanish dictionary for the Spanish support.

9 Contact Details

Dr Nicola Talbot
School of Computing Sciences
University of East Anglia
Norwich. NR4 7TJ.
United Kingdom.
http://theoval.cmp.uea.ac.uk/~nlct/