CNTFORMATS

v0.6   2013/10/11

part of the EXSHEETS bundle

a different way to read counters

Clemens NIEDERBERGER

https://bitbucket.org/cgnieder/exsheets/

contact@mychemistry.eu

Table of Contents

1 Motivation

CNTFORMATS provides a way to format counters with what I will call patterns. This does not in any way effect the usual LATEX 2_ε way of treating counters and does not use \the⟨*counter*⟩ nor is it affected by the redefinition of them.

This package is aimed at package or class authors and probably not very useful for document authors.

When I first had the idea for this package the idea of what it does already existed as part of the EXSHEETS package. I can't recall why I came up with the idea in the first place or why I originally wanted a new syntax for formatting the question counter. I am also not convinced any more that it is a good idea. Anyway, here we are.

2 License and Requirements

Permission is granted to copy, distribute and/or modify this software under the terms of the LATEX Project Public License (LPPL), version 1.3 or later (http://www.latex-project.org/lppl.txt).

The software has the status "maintained."

    CNTFORMATS requires the etoolbox[1] package [Leh11] and the cnltx-base[2] package [Nie14].

3 Example

A use case typically looks as follows:

```
1 \ReadCounterPattern{se.sse}                    3.0
```

where the key se stands for the current value of the section counter and sse for subsection, respectively. se.sse is an example for what will be called *pattern*. The keys for the counters can have optional arguments that specify the format:

```
1 \stepcounter{subsection}                       C(i)
2 \ReadCounterPattern{se[A](sse[r])}
```

A stands for \Alph and r for \roman. A complete list can be found in table 1b on page 5. As you can see you can insert arbitrary other tokens in a pattern that won't be changed. It is important to notice, though, that the patterns are only replaced if they're *not* placed in a braced group!

```
1 \ReadCounterPattern{{se[A](sse[r])}}           se[A](sse[r])
```

    I would imagine that the argument to \ReadCounterPattern is usually supplied by a user setting an option . . .

```
1 \somesetupcommand{
2   counter-format = se[A](sse[r])
3 }
```

. . . and then internally used by the corresponding package or class.

4 Usage

In the following description of the available commands the symbol * means that the command is expandable.

    In order to make counters known to CNTFORMATS the following commands are used:

\AddCounterPattern*[⟨*module*⟩]{⟨*counter*⟩}{⟨*pattern*⟩}
This command will make the (existing) counter ⟨*counter*⟩ known to CNTFORMATS and assign the pattern ⟨*pattern*⟩ to it.

---

1. on CTAN as etoolbox: http://mirrors.ctan.org/macros/latex/contrib/etoolbox/
2. on CTAN as cnltx: http://mirrors.ctan.org/macros/latex/contrib/cnltx/

`\NewCounterPattern`∗[⟨*module*⟩]{⟨*counter*⟩}{⟨*pattern*⟩}

This command will create a new counter ⟨*counter*⟩, make it known to CNTFORMATS and assign the pattern ⟨*pattern*⟩ to it.

`\ReadCounterFrom`[⟨*module*⟩]{⟨*counter*⟩}{⟨*internal cmd*⟩}

If you use one of the commands above with the starred version the number for the pattern is not automatically fetched from the internal `\c@`⟨*counter*⟩. This can now be assigned with `\ReadCounterFrom` where ⟨*internal cmd*⟩ is the macro that holds the number.

The commands above can only be used in the document preamble.

After the creation of these pattern markers one wants to be able to use them. There are a number of macros that allow different aspects of usage.

`\ReadCounterPattern`[⟨*module*⟩]{⟨*pattern*⟩}

Reads, interprets and prints a pattern.

∗ `\@cntfmts@parsed@pattern`

After `\ReadCounterPattern` has been used the current pattern interpretation is stored in this macro. The *interpretation* is *not* what is printed. See the examples below for details.

`\ReadCounterPatternFrom`[⟨*module*⟩]{⟨*macro that holds pattern*⟩}

Reads, interprets and prints a pattern that's stored in a macro.
Otherwise the same as `\ReadCounterPattern`.

`\SaveCounterPattern`{⟨*cmd a*⟩}{⟨*cmd b*⟩}{⟨*pattern*⟩}

Saves the ⟨*pattern*⟩ in ⟨*cmd a*⟩ and the interpreted pattern in ⟨*cmd b*⟩.

`\eSaveCounterPattern`[⟨*module*⟩]{⟨*cmd a*⟩}{⟨*cmd b*⟩}{⟨*pattern*⟩}

Saves the ⟨*pattern*⟩ in ⟨*cmd a*⟩ and the expanded pattern in ⟨*cmd b*⟩.

`\SaveCounterPatternFrom`[⟨*module*⟩]{⟨*cmd a*⟩}{⟨*cmd b*⟩}{⟨*macro that holds pattern*⟩}

Like `\SaveCounterPattern` but reads the pattern from a macro.

`\eSaveCounterPatternFrom`[⟨*module*⟩]{⟨*cmd a*⟩}{⟨*cmd b*⟩}{⟨*macro that holds pattern*⟩}

Like `\eSaveCounterPattern` but reads the pattern from a macro.

The optional argument ⟨*module*⟩ should be specific for a package, say, so that different patterns for the `section` for example don't interfer with each other. If you leave the argument the default module `cntfmts` is used.

The EXSHEETS packages uses the commands with the module `exsheets`. You can find the following lines in EXSHEETS' code:

```
1 \AddCounterPattern*[exsheets]{section}{se}
2 \ReadCounterFrom[exsheets]{section} \l__exsheets_counter_sec_int
3 \NewCounterPattern*[exsheets]{question}{qu}
4 \ReadCounterFrom[exsheets]{question} \l__exsheets_counter_qu_int
```

Now let's see a short example that hopefully explains what the macros do:

```
1  % preamble
2  % \NewCounterPattern{testa}{ta}
3  \setcounter{testa}{11}
4  \ReadCounterPattern{ta}
5  \ReadCounterPattern{ta[a]} \\
6  \ttfamily\makeatletter
7  \meaning\@cntfmts@parsed@pattern
8
9  \bigskip
10 \SaveCounterPattern\tmpa\tmpb{ta[a]}
11 \meaning\tmpa \\
12 \meaning\tmpb
13
14 \bigskip
15 \eSaveCounterPattern\tmpa\tmpb{ta[a]}
16 \meaning\tmpa \\
17 \meaning\tmpb
```

<div style="border-top: 1px solid;"></div>

```
   11 k
macro:->\csuse {@cntfmts@read@ta@counter}[a]\@empty

   macro:->ta[a]
macro:->\csuse {@cntfmts@read@ta@counter}[a]\@empty

   macro:->ta[a]
macro:->k
```

You can see that somehow an additional `\@empty` found its way into the interpreted pattern. This is due to the fact that reading optional arguments expandably isn't easy and must have some safety net.

5 Predefined and New Patterns and Format Keys

5.1 Predefined Patterns and Format Keys

CNTFORMATS predefines a number of pattern keys. These are listed in table 1a.

5.2 New Patterns and Format Keys

Table 1b lists the predefined formats. If you want you can add own formats.

`\NewPatternFormat{⟨pattern⟩}{⟨format⟩}`
⟨*format*⟩ is a number presentation command like `\@alph`, *i. e.*, it needs a mandatory argument that takes a number. It is used in ⟨*format*⟩ *without* its argument. This command can only be used in the preamble.

TABLE 1: Predefined Patterns and Format Keys.

(A) Predefined Patterns for the module `cntfmts`.

counter	pattern
chapter	ch
section	se
subsection	sse
subsubsection	ssse
paragraph	pg

(B) Predefined Format Keys

key	format
1	\arabic
a	\alph
A	\Alph
r	\roman
R	\Roman

Here are now a few examples of possible new patterns. Suppose the following code in the preamble:

```
1  \usepackage{alphalph,fmtcount}
2  \newcommand*\myoddnumber[1]{\the\numexpr2*(#1)-1\relax}
3
4  \NewPatternFormat{aa}{\alphalph}
5  \NewPatternFormat{o}{\ordinalnum}
6  \NewPatternFormat{x}{\myoddnumber}
7
8  \newcounter{testa}
9  \NewCounterPattern{testa}{ta}
10 \setcounter{testa}{4}
```

Then we can use the new pattern and the new formats as follows:

```
1  \ReadCounterPattern{ta[aa]}
2  \ReadCounterPattern{ta[o]}
3  \ReadCounterPattern{ta[x]}
```

d 4th 7

Bibliography

[Leh11] Philipp LEHMAN. etoolbox. version 2.1, Jan. 3, 2011.
URL: http://mirror.ctan.org/macros/latex/contrib/etoolbox/.

[Nie14] Clemens NIEDERBERGER. cnltx. version 0.10a, Jan. 23, 2014.
URL: http://mirror.ctan.org/macros/latex/contrib/cnltx/.

Index