CNTFORMATS

v0.7 2014/07/20

A different way to read counters

Clemens Niederberger

https://github.com/cgnieder/cntformats/

contact@mychemistry.eu

Table of Contents

1 Motivation 1

2 License and Requirements 1

3 Example 2

4 Usage 2

5 Predefined and New Patterns and Format Keys 5

5.1 Predefined Patterns and Format Keys 5

5.2 New Patterns and Format Keys 5

Bibliography 6

Index 7

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 [Nie14b] 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.

Changed in version 0.7

Until and including v0.6 cntformats used to be part of the exsheets bundle. Since version 0.7 it is distributed as a package of its own.

2 License and Requirements

The software has the status “maintained.”

CNTFORMATS requires the etoolbox[1] package [Leh11] and the cnltx-base[2] package [Nie14a].

3 Example

A use case typically looks as follows:

```
\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:

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

`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!

```
\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 …

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

… 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 (and must) 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` counter for example don't interfer with each other. This means: `\ReadCounterPattern` and friends *only* read the patterns known to the specified module! If you leave the argument the default module `cntfmts` is used.

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

```
\AddCounterPattern*[exsheets]{section}{se}
\ReadCounterFrom[exsheets]{section} \l__exsheets_counter_sec_int
\NewCounterPattern*[exsheets]{question}{qu}
```

```
\ReadCounterFrom[exsheets]{question} \l__exsheets_counter_qu_int
```

Another example would be the tasks package [Nie14c]:

```
\NewCounterPattern*[tasks]{task}{tsk}
\ReadCounterFrom[tasks]{task} \g__tasks_int
```

In fact exsheets loads the tasks package and also does the following

```
\AddCounterPattern*[tasks]{question}{qu}
\ReadCounterFrom[tasks]{question}\l__exsheets_counter_qu_int
```

so tasks knows about the question counter and uses the same pattern as exsheets does.

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

```
% preamble
% \NewCounterPattern{test}{t}
\setcounter{test}{11}
\ReadCounterPattern{t}
\ReadCounterPattern{t[a]} \\
\ttfamily\makeatletter
\meaning\@cntfmts@parsed@pattern

\bigskip
\SaveCounterPattern\tmpa\tmpb{t[a]}
\meaning\tmpa \\
\meaning\tmpb

\bigskip
\eSaveCounterPattern\tmpa\tmpb{t[a]}
\meaning\tmpa \\
\meaning\tmpb
```

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

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

  macro:->t[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. In this case looking for an optional argument is done by reading the token following the command as argument. If there is no option then `\@empty` is found so no other tokens

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

provided by users get read which might otherwise lead to strange effects/outcome/errors.[3] On the other hand the grabbing of the next token can lead to spaces after the pattern being ignored, see the next example: if there is no optional argument then the next token is grabbed ignoring the spaces as is normal for a TEX macro. The next example demonstrates this.

A word of caution: although I used a one-letter pattern in the above example I would not recommend this. The pattern should consist of two or more characters otherwise it might get inconvenient hiding the characters that you *don't* want replaced:

```
\setcounter{test}{3}
Where is the first space: \ReadCounterPattern{t t[r] t}?\par
\ReadCounterPattern{every instance of the letter t{} is replaced}
```

Where is the first space: 3iii 3?
every ins3ance of 3he le33er 3 is replaced

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.

3. Imagine for example there would be *no* token following the command – nasty error messages could follow.

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

```
\usepackage{alphalph,fmtcount}
\newcommand*\myoddnumber[1]{\the\numexpr2*(#1)-1\relax}

\NewPatternFormat{aa}{\alphalph}
\NewPatternFormat{o}{\ordinalnum}
\NewPatternFormat{x}{\myoddnumber}

\newcounter{test}
\NewCounterPattern{test}{t}
\setcounter{test}{4}
```

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

```
\ReadCounterPattern{t[aa]}
\ReadCounterPattern{t[o]}
\ReadCounterPattern{t[x]}
```

---

d 4th 7

Bibliography

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

[Nie14a] Clemens Niederberger. cnltx. version 0.10a, Jan. 23, 2014.
URL: http://mirror.ctan.org/macros/latex/contrib/cnltx/.

[Nie14b] Clemens Niederberger. exsheets. version 0.14, June 27, 2014.
URL: http://mirror.ctan.org/macros/latex/contrib/exsheets/.

[Nie14c] Clemens Niederberger. tasks. version 0.10, July 3, 2014.
URL: http://mirror.ctan.org/macros/latex/contrib/tasks/.

Index

Symbols

\@cntfmts@parsed@pattern . . . . . 3 f.

A

\AddCounterPattern . . . . . 2 ff.

C

cnltx (bundle) 2
CTAN 2

E

\eSaveCounterPattern . . . . . 3 f.
\eSaveCounterPatternFrom 3
etoolbox (package) 2
exsheets (bundle) 1
exsheets (package) . . . . . 1, 3 f.

L

Lehman, Philipp 2
LPPL 1

N

\NewCounterPattern . . . . . 3 f., 6
\NewPatternFormat . . . . . 5 f.
Niederberger, Clemens . . . . . 1 f., 4

R

\ReadCounterFrom . . . . . 3 f.
\ReadCounterPattern 2–6
\ReadCounterPatternFrom 3

S

\SaveCounterPattern . . . . . 3 f.
\SaveCounterPatternFrom 3

T

tasks (package) 4