Defining Your Own Macros, Commands and Environments

Ignas Anikevičius

August 29, 2011

Contents

This is one of the most useful TeX features. Actually all LaTeX system is just a set of macros and environments to extend TeX features. Definition of your own commands might look a very difficult thing to do, but actually it is not! And some useful examples will be examined so that you could try to define your own commands *reusing* the code listed in this tutorial and later, writing your own code to get even more elaborate macros.

1 Defining new LaTeX commands

Let's now consider a situation where a person is writing a dissertation on the physical properties of water and he uses its chemical formula (H_2O) a lot. It takes time to typeset such a formula, and the code actually looks like this: `$\mathrm{H_2O}$`. However, if we have something similar to the following code in the preamble, we could just write `\hho`.

```
1 \newcommand{\hho}{\ensuremath{\mathrm{H_2O}}}
```

The best place to get all the necessary knowledge for being able to do the same as was done above would be this page. There you will be able to find simpler examples and much more examples.

Here I will just give you some useful code snippets, which might give you some ideas or turn out to be useful.

1.1 Typesetting units correctly

Well typeset document will usually have very elegant code. Which means, that for example to enter an equation one will not switch back and forth from the `math` mode, but rather enter everything there. However, if units are typeset right away, they might appear in italics (i.e. the same way as the equation variables appear), which is not what we want.

There are several packages which might be worth looking, but I find it just unnecessary as one can solve this issue with only one short command.

Let's consider a following expression:

$$m = \rho \times V = 1000\,\mathrm{kg}\cdot\mathrm{m}^{-3} \times 3\,\mathrm{m}^{-3} = 3000\,\mathrm{kg}$$

and the code for it can be found bellow:

```
\[
    m = \rho \times V
      = 1000 \, \mathrm{kg \cdot m^{-3}}
      \times 3 \, \mathrm{m^{-3}}
      = 3000 \, \mathrm{kg}
\]
```

Now look at the equation generated by another version of the code where we do not enter all those tedious commands:

$$m = \rho \times V = 1000\,\mathrm{kg}\cdot\mathrm{m}^{-3} \times 3\,\mathrm{m}^{-3} = 3000\,\mathrm{kg}$$

```
\[
    m = \rho \times V
      = 1000 \un{kg \cdot m^{-3}}
      \times 3 \un{m^{-3}}
      = 3000 \un{kg}
\]
```

This can be achieved with the following command in the preamble of the file.

```
\newcommand{\un}[1]{\ensuremath{\, \mathrm{#1}}}
```

You might wonder, why to use a newly defined command, if the amount of code does not decrease very much. However, you **should** think about the case where you would like to change the formatting of the units just ever so slightly and if you used the `\un` command, which you had defined previously, you would need to change only one line, but otherwise you would need to change the entire document, which I believe is not the fastest way to do it.

1.2 Typesetting lengthy and tedious expressions quickly

This probably might apply more to people who are dealing with uncertainties more and need to write expressions which are quite repetitive in a sense. Here is an example with relative fractions.

Suppose we have the following the equations:

$$p = \frac{mRT}{MV}$$
$$\frac{\Delta p}{p} = \frac{\Delta m}{m} + \frac{\Delta T}{T} + \frac{\Delta V}{V}$$

Please notice, that the relative errors do have very repetitive expressions, as we always have a fraction and the symbol is repeated twice in the numerator and denumerator, just on the numerator there is an extra Δ symbol in front.

Now let's compare two possible ways to achieve the above line for the fractional errors:

```
\begin{align*}
    \cfrac{\Delta p}{p} &= \cfrac{\Delta m}{m} + \cfrac{\Delta T}{T} +
        \cfrac{\Delta V}{V}
\end{align*}
```

and

```
\begin{align*}
    \rel{p} &= \rel{m} + \rel{T} + \rel{V}
\end{align*}
```

After comparing the two code listings it is evident, that the source code of the second variant is much easier to comprehend and read than the first one. And it is achieved by defining the `\rel` command in the preamble:

```
\newcommand{\rel}[1]{\ensuremath{\cfrac{\Delta #1}{#1}}}
```

2 Counters and more elaborate commands

Since we now how to define new commands in LaTeX and ease our life in this way, we can now go one step further. Suppose we have a large document with a lot of figures and in those figures we have a lot of structures, which we would like to number and refer easily in the text.

2.1 Example 1: Single Level Counters (1,2,3...) for Chemical Structures

So defining compound **1** and **2** is very easy. Referencing them (Compound 2 and 1) is also easy. Look at the figure 1 to see how you can incorporate it into figures.

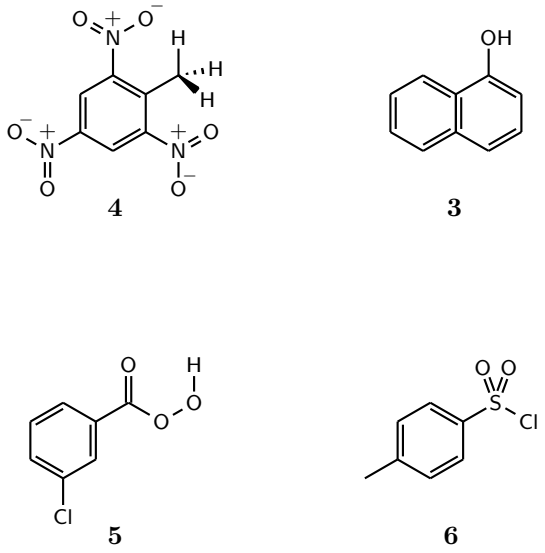

Figure 1. Using the the newly-created `\cmp` command in figures. Although this thing was done using the `TikZ` package, the same would work with other methods, such as overpic package or others, it is just that the alignments of the labels might be somewhat more difficult.

The code required to get the compound numbering is shown bellow. Note that the structures were made using the `ChemFig` package, whereas the image was composed with the `TikZ` package. The code required to draw the structures is available in the `compnum-tikz.tex` file supplied together in the source archive.

```
\begin{figure}[h]
    \centering
    \begin{tikzpicture}
        ...
        \draw (naphtol.base)
            node[label,yshift=-1cm]{\cmp{cmp:naphtol}};
        \draw (TNT.base)        node[label,yshift=-1cm]{\cmp{cmp:TNT}};
        \draw (mCPBA.base)      node[label,yshift=-1cm]{\cmp{cmp:mCPBA}};
        \draw (TsCl.base)       node[label,yshift=-1cm]{\cmp{cmp:TsCl}};
    \end{tikzpicture}
```

```
    ...
\end{figure}
```

Note that the number are assigned in the order of the command execution. Hence, it really does not matter where the number is located in the figure as long as the labels are assigned in the correct order.

The `\cmp` command definition is as follows (Note the comment characters after the commands, which are used in order to avoid unwanted spaces in the final result):

```
% Initial and format the counter.
\newcounter{chemcmp}
\renewcommand{\thechemcmp}{\arabic{chemcmp}}
%
% Command for numbering compounds like 1,2,3,...
\newcommand{\cmp}[1]{%
\refstepcounter{chemcmp}%
\textbf{\thechemcmp}%
\label{#1}}
```

2.2 Example 2: Two Level Counters (1a,1b,2,3...) for Chemical Structures

Now we can extend the idea of numbering the compounds and including numbers like 1a, 1b. I will use the same for structures, just I will change the numbers.

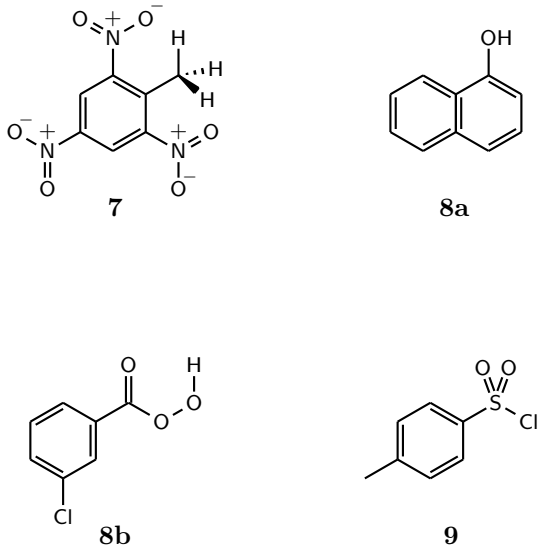

Figure 2. The same example as previously, but with more elaborate numbering command.

The code bellow is for the second example. Note that the syntax of the commands has changed. That is because we need one command to increment the parent counter (i.e. going 1,2,3,...) and we need another command to change child counter (i.e. 1a,1b,1c,...), however, we need one more command to change both counters (i.e. 1e $\longrightarrow$ 2a...). The same structures where used and nothing was change in terms of drawing.

```
\begin{figure}[h]
    \centering
    \begin{tikzpicture}
        ...
        \draw (TNT.base)        node[label,yshift=-1cm]{\cmp{cmp:TNT2}};
```

```
        \draw (naphtol.base)
            node[label,yshift=-1cm]{\cmppn{cmp:naphtol2}};
        \draw (mCPBA.base)
            node[label,yshift=-1cm]{\cmppe{cmp:mCPBA2}};
        \draw (TsCl.base)        node[label,yshift=-1cm]{\cmp{cmp:TsCl2}};
    \end{tikzpicture}
    ...
\end{figure}
```

The `\cmp` name stands for the 'compound' whereas `\cmppn` is for 'sub-compound new' and `\cmppe` is for 'sub-compound existing'. The reason why I have made 2 commands for 'sub-compounds' is that one is for the entry of the first 'sub-compound' (e.g. 1a) and the other compound is for adding 'sub-compounds' (e.g. 1b, 1c, 1d...). The code in the preamble defining the commands `\cmp`, `\cmppn`, `\cmppe` is as follows:

```
% Initialize 2 additional counters.
\newcounter{chemcmp}
\newcounter{chemcmpp}
%
% Set the formatting of the counters.
\renewcommand{\thechemcmp}{\arabic{chemcmp}} % The parent counter will
\renewcommand{\thechemcmpp}{\alph{chemcmpp}} % use arabic numerals
                                             % whereas the child
                                             % counter will use latin
                                             % letters.
% Command to increment the parent counter
\newcommand{\cmpinc}{%
\refstepcounter{chemcmp}%
\setcounter{chemcmpp}{0}}
%
% Command for 1 -> 2 transition
\newcommand{\cmp}[1]{\cmpinc%
\textbf{\thechemcmp}%
\label{#1}}
%
% Command for 1b -> 1c transition
\newcommand{\cmppe}[1]{%
\refstepcounter{chemcmpp}%
\textbf{\thechemcmp\thechemcmpp}%
\label{#1}}
%
% Command for 1x -> 2a transition.
\newcommand{\cmppn}[1]{\cmpinc%
\refstepcounter{chemcmpp}%
\textbf{\thechemcmp\thechemcmpp}%
\label{#1}}
```

2.3 Example 3: How easy it is to change the format of numbering

Since we have defined quite powerful macros for numbering compounds, it would be nice to show you how easy it is to change the formatting of the numbering. By executing the following commands I will reset both counters back to zero, so that the compounds would get numbered from 1 all over again. However, the second command will change the child counter to appear not as Latin symbols, but rather as lower-case Roman numerals.

```
\setcounter{chemcmp}{0}          % Reset both counters
\setcounter{chemcmpp}{0}
\renewcommand{\thechemcmpp}{(\roman{chemcmpp})} % use
    1(i),1(ii),1(iii),...
```

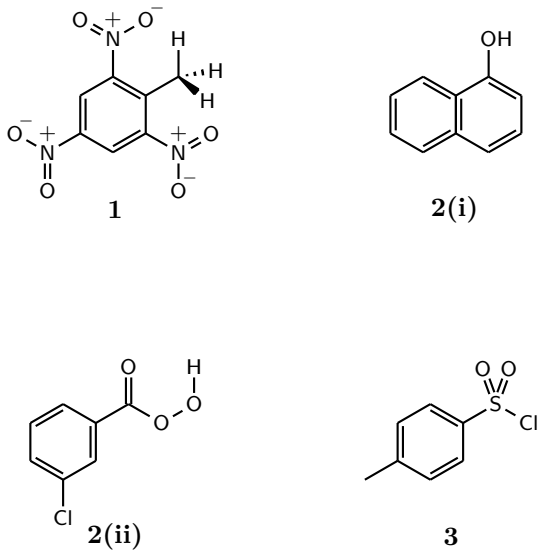

Figure 3. The same example as previously, but with reset counters and different child counter formatting.

Now it becomes evident, that with LaTeX macros the formatting of the content becomes a matter of alteration of the macros themselves and not changing the text, which might take longer and one might make more mistakes in changing it.