# Defining your own macros, commands and environments

Ignas Anikevicius

August 24, 2011

## Contents

## Todo list

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! I will examine some examples in the following sections

Update the source code, so that it would be nicely indented.

## 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`.

```
\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 (ie. 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:

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

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:

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

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

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:

$$\begin{aligned} p &= \frac{mRT}{MV} \\ \frac{\Delta p}{p} &= \frac{\Delta m}{m} + \frac{\Delta T}{T} + \frac{\Delta V}{V} \end{aligned}$$

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 $\Delta$ 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*}
```

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

I guess you can see how much more readable the second variant is. 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.

update the examples so that ChemFig would be used instead of msketch figures

### 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.

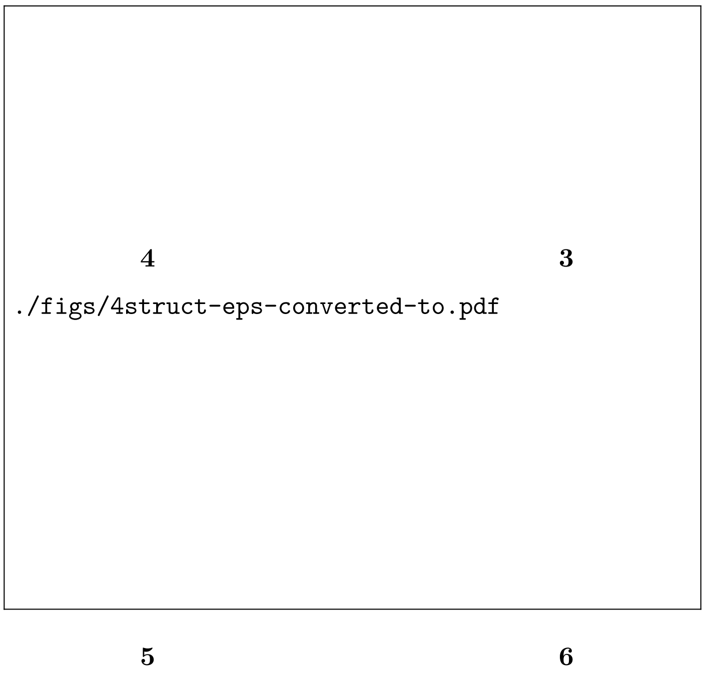

**Figure 1.** Overlaying LaTeX commands on top of the figure. To get the correct numbers we used a newly created `cmp` command. As you see the order of the numbers is determined by the order of code execution.

```
So defining compound \cmp{cmp:benzene} and \cmp{cmp:naphtalene} is
very easy.  Referencing them (Compound \ref{cmp:naphtalene} and
\ref{cmp:benzene}) is also easy.
Look at the figure \ref{fig:compeg} to see how you can incorporate it
into figures.

\begin{figure}[h]
```

```
    \centering
    \setlength{\tikzunit}{.085\textwidth}
    \begin{tikzpicture}[scale=1.0,x=\tikzunit,y=\tikzunit]
%----- Draw a grid which should help to position things ------------
%         \draw[step= .5,color=gray,thin,dashed] (-4,-4) grid (4,4);
%         \draw[step=1.0,color=gray]             (-4,-4) grid (4,4);
%         \draw[step=4.0,color=black]            (-4,-4) grid (4,4);
%  Notes:
%       just uncomment the lines with draw commands and the grid
%       will appear.  The commands, I believe are self explanatory
%       and it can be drawn as big as you want. The two coordinates
%       denote lower left and upper right corners of the grid.
%--------------------------------------------------------------------
        \node(0,0){\includegraphics[width=7\tikzunit]{4struct.eps}};
        \draw ( 2.1,  0.5) node{\cmp{cmp:naphtol}};
        \draw (-2.1,  0.5) node{\cmp{cmp:TNT}};
        \draw (-2.1, -3.5) node{\cmp{cmp:mCPBA}};
        \draw ( 2.1, -3.5) node{\cmp{cmp:TsCl}};
    \end{tikzpicture}
    \caption{Overlaying \LaTeX{} commands on top of the figure. To
    get the correct numbers we used a newly created
    \texttt{cmp} command. As you see the order of the numbers is
    determined by the order of code execution.}
    \label{fig:compeg}
\end{figure}
```

The `\cmp` command definition is as follows:

```
\newcounter{chemcmp}
\renewcommand{\thechemcmp}{\arabic{chemcmp} }
\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.

```
\begin{figure}[h]
    \centering
    \setlength{\tikzunit}{.085\textwidth}
    \begin{tikzpicture}[scale=1.0,x=\tikzunit,y=\tikzunit]
%----- Draw a grid which should help to position things ------------
%         \draw[step= .5,color=gray,thin,dashed] (-4,-4) grid (4,4);
%         \draw[step=1.0,color=gray]             (-4,-4) grid (4,4);
%         \draw[step=4.0,color=black]            (-4,-4) grid (4,4);
%  Notes:
%       just uncomment the lines with draw commands and the grid
%       will appear.  The commands, I believe are self explanatory
%       and it can be drawn as big as you want. The two coordinates
%       denote lower left and upper right corners of the grid.
%--------------------------------------------------------------------
        \node(0,0){\includegraphics[width=7\tikzunit]{4struct.eps}};
        \draw (-2.1,  0.5) node{\cmp{cmp:TNT1}};
```

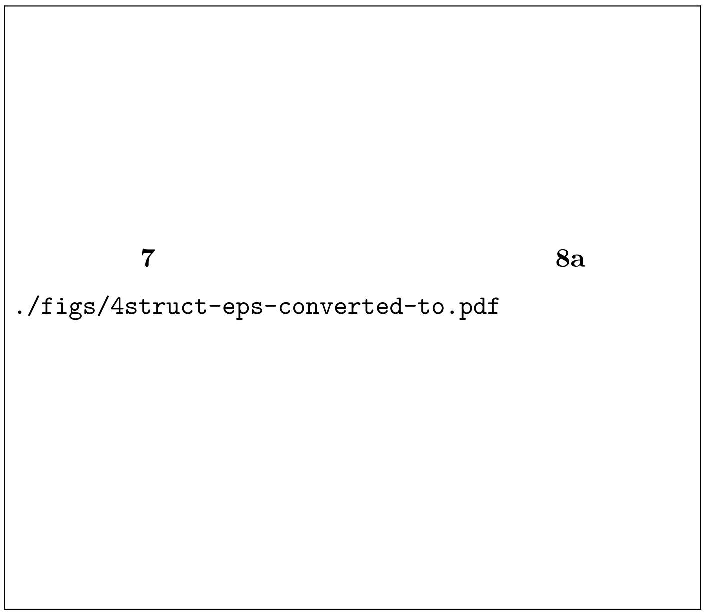

**Figure 2.** Overlaying LaTeX commands to get some nice things.

```
          \draw (  2.1,  0.5) node{\cmppn{cmp:napht-2-ol1}};
          \draw (-2.1,-3.5) node{\cmppe{cmp:mCPBA1}};
          \draw (  2.1,-3.5) node{\cmppe{cmp:TsCl1}};
     \end{tikzpicture}
     \caption{Overlaying \LaTeX\ commands to get some nice things.}
     \label{fig:compegg}
\end{figure}
```

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

```
\newcounter{chemcmp}
\newcounter{chemcmpp}
\renewcommand{\thechemcmp}{\arabic{chemcmp}}
\renewcommand{\thechemcmpp}{\alph{chemcmpp}}
\newcommand{\cmpinc}{
     \refstepcounter{chemcmp}
     \setcounter{chemcmpp}{0}
}
\newcommand{\cmp}[1]{
     \cmpinc
     \textbf{\thechemcmp }
     \label{#1}
}
\newcommand{\cmppe}[1]{
     \refstepcounter{chemcmpp}
     \textbf{\thechemcmp\thechemcmpp }
```

```
        \label{#1}
}
\newcommand{\cmppn}[1]{
        \cmpinc
        \refstepcounter{chemcmpp}
        \textbf{\thechemcmp\thechemcmpp }
        \label{#1}
}
```