LaTeX Customization and Tips

Viktor Dmitriyev

Adapter from Mini Course on LaTeX by David Diez

Outline

- Mathematics in LaTeX
- BibTeX: bibliographies in LaTeX
- Building your own commands and environments
- Miscellaneous tips

Guide to LaTeX

The book *Guide to LaTeX* offers a very nice introduction, and we will closely follow some of the examples in these chapters in this class:

7 math

11,12 BibTeX, and

10 custom commands and environments.

Custom command material

- Counters
- Creating commands
- Creating environments

Existing counters

LaTeX uses counters (variables) to keep proper numbering.

- The following are counters used by LaTeX: `part`, `chapter`, `section`, `subsection`, `subsubsection`, `page`, `footnote`, `equation`, `figure`, and `table`. These counters correspond to their corresponding commands.
- Other counters are used for each level of enumerate: `enumi`, `enumii`, `enumiii`, `enumiv`.
- A few other LaTeX counters: `paragraph`, `subparagraph`, `mpfootnote`.

Create new counters

We may want to create our own counters for our own purposes. Maybe we have examples that we want numbered.

```
\newcounter{counterName}[inCounter]
```

This creates a new counter called `counterName`. The `inCounter` argument + brackets are optional and an `inCounter` is used to reset `counterName` whenever `inCounter` increments (e.g. `subsection` has "inCounter" `section`).

Modifying

We can modify existing or new counters.

\setcounter{counter}{n}

\addtocounter{counter}{n}

\stepcounter{counter}

\refstepcounter{counter}

The \refstepcounter command lets us reference our counter value if we follow it with a \label.

Printing

So we can create, modify, and reference counters. However, we also need to print counters in the document. We do so by calling the counter with one of the following commands:

- \arabic{chapter} (4, Arabic number)
- \Roman (IV, uppercase Roman numeral)
- \roman (iv, lowercase Roman numeral)
- \Alph (D, capital letter)
- \alph (d, lowercase letter)
- \fnsymbol (§, footnote symbol)

We will put counters to use in our custom commands and environments.

Simple command

Common statements, like $x_1, ..., x_n$ can be abbreviated using a new command.

```
\newcommand{\xvec}{x_1,\dots,x_n}
```

Inserting this command and then typing (later in the document) `$\xvec$`, we get $x_1, \dots, x_n$. If we forgot the dollar signs, we would be in trouble. We can resolve this by using an extra command:

```
\newcommand{\xvec}{\ensuremath{x_1,\dots,x_n} }
```

In the second definition, we left an extra space at the end, which helps prevent spacing problems. More elegant solution is to use the xspace package (see *Guide to LaTeX*, page 186).

Command with arguments

If we want to generalize our command, we add two arguments

```
\newcommand{\subvec}[2]{\ensuremath{#1_1,\dots,#1_{#2}}
}
```

We can create $y_1, \dots, y_m$ from `\subvec{y}{m}`.

Additional arguments can be created and are referenced via #n for the n^{th} argument. Optional default arguments can also be utilized (see *Guide to LaTeX*, page 188).

Generalization

The general framework of new commands is

\newcommand{\commandName}[n]{the commands}

where

- `commandName` is the name of the command,
- `n` is the number of arguments, and
- the arguments are referred to as #1, #2, . . . , #n in the commands.

To redefine a command that already exists, use \renewcommand with the same format as above.

Sample environment

Environments use begin and end tags (e.g. `itemize`). We only need define what happens at the \begin and \end tags. For example,

```
\newenvironment{example}
 {\small\textbf{Example.} \hspace{2mm}} % begin stuff
 {\\} % end stuff
```

Sample environment call:

```
\begin{example}
Modular addition works in mysterious ways: $2+2=1$ (mod 3).
\end{example}
```

Result:

Example. Modular addition works in mysterious ways: $2 + 2 = 1$ (mod 3).

General environment

Generally environments take the form

```
\newenvironment{environmentName}{begin stuff}{end stuff}
```

We can also declare that there will be `n` arguments.

```
\newenvironment{environmentName}[n]{begin stuff}{end
stuff}
```

As before, we refer to the arguments as #1, . . . , #n in the begin stuff and end stuff.

To redefine an environment that already exists, use `\renewenvironment` with the same format as above.

Environment + counter

```
\newcounter{example}
\setcounter{example}{0}
\newenvironment{example}
  {\refstepcounter{example}\small
    \textbf{Example \arabic{example}.}\hspace{2mm}}
  {\\}
```

Organizer and time saver

The \include command is useful for long documents:

```
\include{otherDocName}
```

For instance, this presentation actually calls three separate documents: one for each big section. Thus I would not take time Typesetting parts of the document I was not working on while keeping organized:

```
\include{tips/tips} % “tips” document in the “tips” folder
```

Summary

After this class, you should have a general idea of

- creating your own commands and environments and
- random tips.

Any questions?