Using Markdown inside TEX documents

Vít Novotný

Abstract

Markdown is a lightweight markup language that makes it easy to write structurally simple documents using a clean and straightforward syntax. Although various tools for rendering Markdown documents via TEX exist, they tend to be built on top of TEX rather than in TEX.

This paper briefly presents existing tools and introduces a macro package for plain-based TEX formats that takes a different approach. By making it possible to put snippets of Markdown-formatted text into arbitrary TEX documents and exposing TEX macros that control the rendering of Markdown elements, the package provides a convenient way of introducing Markdown into existing TEX workflows.

1 Introduction

TEX is a fine tool for typesetting many kinds of documents. It may, however, not always be the best language for writing them. Markup languages based on SGML and XML make it possible for an author to focus on the content of their documents without having to worry about the error messages produced by commands and unforeseen macro package interactions. The resulting documents can then be transformed not only to various TEX formats, but also to other output formats that bear no relation to the TEX world.

When preparing structurally simple documents, however, SGML and XML with their bulky syntax may feel too heavy-handed. For these kinds of documents, lightweight markup languages that exchange raw expressive power for clean and simple syntax are often the best choice. In this paper, I will focus on the lightweight markup language of Markdown [2].

Although the language of Markdown was originally envisioned as an HTML preprocessor, its syntax is agnostic to the output format, which makes Markdown useful as a general document markup language. Tools that provide conversion from Markdown to various TEX formats are therefore readily available. One of the better-known free open-source programs that enable conversion from Markdown to TEX is Pandoc [4]. Dubbed a Swiss Army Knife by its author, Pandoc enables the conversion between an impressive number of markup languages (e.g. LATEX, ConTEXt, HTML, XML Docbook) and output formats (e.g. ODF, OOXML, or PDF). The tool has already been reviewed in *TUGboat* by Massimiliano Dominici [1].

Pandoc is a powerful multi-target publishing software and its ability to perform lossy conversions (such as from LaTeX to HTML) makes it extremely useful for document manipulation in general. However, if our sole goal is to use Markdown markup inside TeX documents, Pandoc displays several weaknesses.

The ability to redefine the correspondence between Markdown elements in the input and TeX macros in the output is limited. Processing the following Markdown document:

```
- Single underlines/asterisks denote _emphasis_.
- Double them for **strong emphasis**.
- The *two* __may be__ _freely *mixed*_.
```

in Pandoc v1.17.2 using the command `pandoc -f markdown -t latex` produces the following LaTeX document:

```
\begin{itemize}\tightlist
\item Single underlines/asterisks denote
     \emph{emphasis}.
\item Double them for \textbf{strong emphasis}.
\item The \emph{two} \textbf{may be}
     \emph{freely \emph{mixed}}.
\end{itemize}
```

While it is possible to redefine the produced LaTeX macros in theory, altering base macros such as `\item` or `\textbf` may break the document in subtle ways. The output is also not fixed and may vary between different versions of Pandoc.

Another desirable feature is sandboxing. Markdown is a static markup language without programming capabilities and may be used by ordinary users without much training. If Markdown documents are submitted to a system and then automatically typeset using TeX, then these documents should certainly not be able to crash or halt the compilation, or to execute external programs using the `\write18` command and similar mechanisms. Pandoc does not offer much in these regards, since it permits TeX macros in the input Markdown documents. There exist complex rules for deciding whether or not an occurrence of a TeX special character should be kept or removed; the following document:

```
This {will} 2^n \begin{get} r~moved and \this
{won't} \begin{equation}2^n\end{equation}$2^n$.
```

when converted with Pandoc becomes:

```
This \{will\} 2\^{}n \textbackslash{}begin\{%
get\}r\textasciitilde{}moved and \this{won't}
\begin{equation}2^n\end{equation} \(2^n\).
```

Apparently, the aim is to enable the use of mathematics and simple TeX macros while retaining baseline compatibility with standard Markdown documents that may contain portions resembling plain TeX. As a result, users are limited in their ability to use TeX inside their Markdown documents, but there is still plenty of rope left for halts, crashes, and external command execution.

Another inconvenience of Pandoc is its lack of integration with the TeX distributions. TeX documents without external dependencies and written in stable formats such as plain TeX require virtually no maintenance. The use of external assets and actively developed formats such as ConTeXt will require some attention each time there is a major TeX distribution release. Software outside TeX distributions such as Pandoc throws more variables into the mix, since different versions may produce different output even if the TeX distribution stays the same. Besides the trickier maintenance, Pandoc's absence from TeX distributions also means that it is unavailable in major TeX services such as the collaborative text editors at `http://www.sharelatex.com/` and `http://www.overleaf.com/`.

Other major tools for rendering Markdown, such as MultiMarkdown, were reviewed and found to be plagued by similar design choices. With this knowledge, I decided to prepare a macro package for rendering Markdown inside TeX that would take a different approach. The goals were:

- to make it easy to specify how individual Markdown elements should be rendered,
- to provide sandboxing capabilities, and
- to make sure that the package required nothing more than what was present in standard TeX distributions.

2 The markdown.tex package

2.1 Architectural overview

The block diagram in figure 1 summarizes the high-level structure of the package. Working from the bottom of the diagram upwards, I will now describe the individual layers.

The translation from Markdown to TeX is done by the Lunamark Lua library [3]. The library was modified, so that it would not depend on external libraries and so that it would produce an intermediate plain TeX representation of the input Markdown document. This means that instead of Pandoc's TeX representation (repeated here from the previous page for ease of reference),

```
\begin{itemize}\tightlist
\item Single underlines/asterisks denote
     \emph{emphasis}.
\item Double them for \textbf{strong emphasis}.
\item The \emph{two} \textbf{may be}
     \emph{freely \emph{mixed}}.
\end{itemize}
```

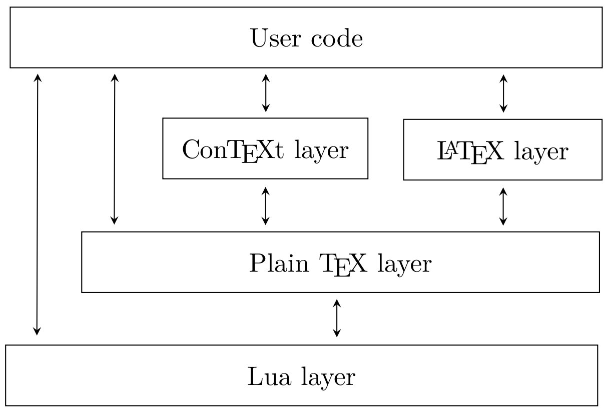

Figure 1: A block diagram of the package

the library will produce the following representation:

```
\markdownRendererUlBeginTight
    \markdownRendererUlItem
        Single underlines/asterisks denote
        \markdownRendererEmphasis{emphasis}.
    \markdownRendererUlItemEnd
    \markdownRendererUlItem
        Double them for
        \markdownRendererStrongEmphasis{strong
        emphasis}.
    \markdownRendererUlItemEnd
    \markdownRendererUlItem
        The \markdownRendererEmphasis{two}
        \markdownRendererStrongEmphasis{may be}
        \markdownRendererEmphasis{freely
        \markdownRendererEmphasis{mixed}}.
    \markdownRendererUlItemEnd
\markdownRendererUlEndTight
```

This representation has two useful properties: it works with any TEX format that uses the same specials as plain TEX does and it represents the logical structure of the input Markdown document in terms of macros that can be freely redefined by the user.

The plain TEX layer exposes the capabilities of the Lua library as TEX macros and provides default definitions for the `\markdownRenderer`⟨*Name*⟩ macros from the intermediary representation. Macro package developers are encouraged to redefine the `\markdownRenderer`⟨*Name*⟩`Prototype` macros that correspond to the default definitions. When the LuaTEX engine is used, the Lua library is accessed directly. Otherwise, the shell escape (`\write18`) mechanism is used (see [5, sec. 3.2.5] for details).

The LATEX and ConTEXt layers correct for some idiosyncrasies of the respective TEX formats and provide user-friendly variants of several macros from the plain TEX layer and sane default definitions for the `\markdownRenderer`⟨*Name*⟩ macros. Developers are encouraged to contribute layers for other formats.

2.2 Usage examples

This section contains examples for markdown.tex (version 2.5.3). These should give an idea of the capabilities of the package. The examples are in LATEX for ease of exposition. As noted in the previous section, the LATEX layer of the package is reasonably thin and the examples can therefore be easily adapted for the plain TEX and ConTEXt layers. For brevity, the examples will contain only the body of a LATEX document assuming the following preamble:

```
\documentclass{article}
\usepackage{markdown,graphicx}
```

To typeset the examples, you can use the `lualatex` or `pdflatex -shell-escape` commands. For further information, see the package documentation [5].

Sandboxing disables hybrid markup and is enabled by default. As a result, the following example:

```
\begin{markdown}
$\sqrt{x^2 + y^2}$
\end{markdown}
```

will produce $\sqrt{x^2 + y^2}$. To enable the use of hybrid markup, the `hybrid` option needs to be specified. The following example:

```
\begin{markdown*}{hybrid}
$\sqrt{x^2 + y^2}$
\end{markdown*}
```

will produce $\sqrt{x^2+y^2}$ as expected. You may also create a partial sandbox; the following example enables the use of non-breaking spaces:

```
\begin{markdown*}{renderers={tilde=~}}
Bartel~Leendert van~der~Waerden
\end{markdown*}
```

With hybrid markup, using underscores and backticks may produce unexpected results. That is because Markdown uses underscores for emphasis and backticks for inline verbatim text, whereas LATEX uses underscores for math subscripts and backticks for opening quotation marks. Preceding characters with a backslash disables their special meaning in Markdown, as the following example shows:

```
\begin{markdown*}{hybrid}
\`\`This is a quote with $a\_{subscript}$.''
\end{markdown*}
```

This, however, makes the text difficult to both read and write. Alternatively, you can disable backticks and underscores in Markdown, as the following example shows:

```
\begin{markdown*}{hybrid, codeSpans=false,
                  underscores=false}
``This is a quote with $a_{subscript}$.''
\end{markdown*}
```

Emphasis can still be denoted via asterisks.

Since the standard Markdown syntax covers only the essentials, the package supports a number of syntax extensions that allow you to mark up moderately complex content without hybrid markup. The following example gives a taste of what is available:

```
\begin{markdown*}{citations, contentBlocks,
                  footnotes, inlineFootnotes}
[^1] and ^[inline footnotes] are highly useful,
as shown in the table below.

  /table.csv

The table was borrowed from @doe12 [p. 34].

[^1]: Footnotes
\end{markdown*}
```

See [5, sec. 2.1.2] for the full list of syntax extensions.

The Markdown syntax permits online images, but the package currently does not handle these in any special way. Therefore, if you would like to download and typeset online images, you will need to provide your own implementation. One such implementation is shown in the following example:

```
\begingroup
\catcode‘\@=11
\catcode‘\%=12
\catcode‘\^^A=14
\global\def\markdownRendererImage#1#2#3#4{^^A
\immediate\write18{^^A
  if printf ’%s’ "#3" | grep -q ^http; then
    OUTPUT="$(printf ’%s’ "#3" | md5sum |
              cut -d’ ’ -f1).^^A
            $(printf ’%s’ "#3" |
              sed ’s/.*[.]//’)";
    if ! [ -e "$OUTPUT" ]; then
      wget -O "$OUTPUT" ’#3’ || rm "$OUTPUT";
      convert "$OUTPUT" png:"$OUTPUT";
    fi;
    printf ’%s%%’ "$OUTPUT" > \jobname.fetched;
  else
    printf ’%s%%’ "#3"      > \jobname.fetched;
  fi}^^A
{\everyeof={\noexpand}^^A
\edef\filename{\@@input"\jobname.fetched" }^^A
\includegraphics[width=\textwidth]{\filename}}}
\endgroup

\begin{markdown}
![TUGboat](https://tug.org/tugboat/noword.jpg)
The Communications of the TeX Users Group
\end{markdown}
```

Note that this implementation expects a Unix-like operating system with a Bourne-compatible shell. It also assumes that the md5sum, wget, and convert binaries are installed and that the TeX engine has shell access, among other things.

3 Conclusions

With the new markdown.tex package, it is now possible to typeset Markdown documents in TeX without the need for external tools. This notably enables the use of Markdown in collaborative text editors such as http://www.sharelatex.com/ and http://www.overleaf.com/, and in other services where tools from outside TeX distributions are unavailable.

The package also gives the authors full control over how individual Markdown elements are rendered and how much access to TeX markup the Markdown documents have. The former encourages creative domain-specific use of the Markdown syntax and the latter enables the use of TeX for the unsupervised typesetting of user-submitted Markdown documents.

Acknowledgements

I gratefully acknowledge the funding received from the Faculty of Informatics at the Masaryk University in Brno for the development of the package.

I would also like to thank John MacFarlane, the creator of the Lunamark Lua library, for having released Lunamark under a permissive license that enabled its inclusion into the package.

References

[1] Massimiliano Dominici. An overview of Pandoc. *TUGboat*, 35(1):44–50, 2014. http://tug.org/TUGboat/tb35-1/tb109dominici.pdf (visited on 2016-08-15).

[2] John Gruber. Daring Fireball: Markdown, 2013. http://daringfireball.net/projects/markdown/ (visited on 2016-08-15).

[3] John MacFarlane. Lunamark, 2012. http://jgm.github.io/lunamark/doc/ (visited on 2016-08-17).

[4] John MacFarlane. Pandoc: A universal document converter, 2016. http://pandoc.org/ (visited on 2016-08-15).

[5] Vít Novotný. *A Markdown Interpreter for TeX*, 2017. https://ctan.org/pkg/markdown (visited on 2017-04-10).

◇ Vít Novotný
Nad Cihelnou 602
Velešín, 382 32
Czech Republic
witiko (at) mail dot muni dot cz
https://github.com/witiko