Markdown 2.7.0: Towards lightweight markup in TeX

Vít Novotný

Abstract

Markdown is a lightweight markup language that makes it easy to write structurally simple documents. Existing tools for rendering markdown documents to PDF treat TeX as a black box. In contrast, the Markdown package provides support for styling and typesetting markdown documents in TeX, extending a TeXie's toolbox rather than forcing her to replace TeX with a more limited tool.

Since its release in 2016, the package has received several important updates improving the functionality and user experience. In this article, I will reintroduce the package, and describe its new functionality and documentation.

1 Introduction

The primary strength of TeX lies perhaps in its programming and typesetting capabilities, not its syntax. Outside mathematics, non-programmable markup languages such as Markdown [2] provide a gentler learning curve, improved readability, and effortless single-source publishing for an author.

Existing tools for rendering markdown documents to PDF, such as Pandoc [1, 3] and MultiMarkdown, have several important disadvantages, which I discussed in my previous article [5]. These disadvantages include black-boxing TeX, inconsistent support for TeX commands in markdown documents, increased complexity of maintenance, and the lack of support for online TeX services such as Overleaf. The Markdown TeX package [5] overcomes all of these.

In my previous article, I introduced version 2.5.3 of the Markdown package, which was plagued by several shortcomings: The package would not function correctly when the `-output-directory` TeX option was specified, since the package interacts with an external Lua interpreter that is unaware of TeX options. The package was also wasteful with system resources, clogging up the file system with converted markdown documents unless the user cleaned them up manually. The documentation of the package was complete, but provided little help to the non-technical user.

In this article, I introduce version 2.7.0 of the Markdown package, which tackles the above problems and introduces several new features, such as content slicing and the Lua CLI. In Section 2, I will show the new features. In Section 3, I will describe the new documentation of the package.

2 New features

Between versions 2.5.3 and 2.7.0 of the Markdown package, there was one important patch version, 2.5.4, and two important minor versions, 2.6.0 and 2.7.0. Version 2.5.4 added support for the TeX option `-output-directory`, version 2.6.0 introduced the Lua command-line interface (CLI) and added support for the `doc` LaTeX package [4], and version 2.7.0 introduced the user manual and content slicing.

In this section, I will show the new features. Although the package also supports plain TeX and ConTeXt, all examples are in LaTeX for ease of exposition.

2.1 Setting the output directory

TeX provides the `-output-directory` option, which changes the working directory for the TeX document. This allows the user to redirect auxiliary files to a separate location. However, any external programs executed using the `\write18` mechanism run in the original working directory. Since the Markdown package executes a Lua interpreter, which expects to find auxiliary files produced by the package in the current working directory, this presents a problem.

To solve the problem, version 2.5.4 of the Markdown package introduced the new `outputDir` option, which informs the Lua interpreter where it should look for the auxiliary files. Create a text document named `document.tex` with the following content:

```
\documentclass{article}
\usepackage[outputDir=/dev/shm]{markdown}
\begin{document}
\begin{markdown}
A First Level Header
====================
A Second Level Header
---------------------
Now is the time for all good men to come to the
aid of their country. This is just a paragraph.
\end{markdown}
\end{document}
```

Execute the following command to produce a document with a single section, subsection, and paragraph, redirecting auxiliary files to `/dev/shm` (the RAM disk available on recent Linux kernels):

```
pdflatex -output-directory /dev/shm \
         -shell-escape document.tex
```

2.2 The Lua command-line interface

The Markdown package hands markdown documents to a Lua parser. The parser converts them to TeX and hands them back to the package for typesetting (see Figure 1). This procedure has the advantage of being fully automated. However, it also has several

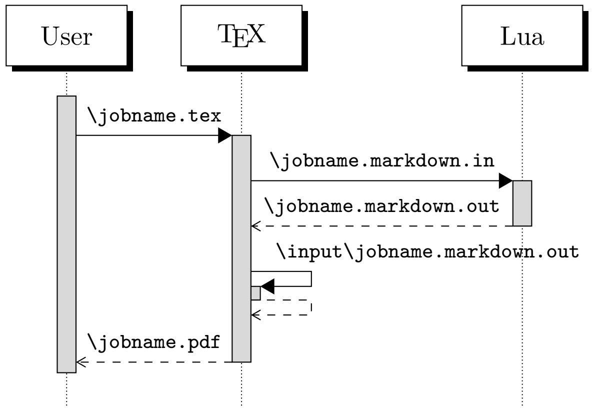

Figure 1: A sequence diagram of the Markdown package typesetting a markdown document using the TEX interface.

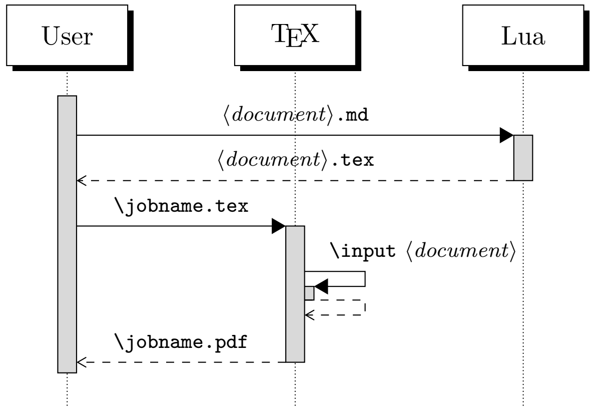

Figure 2: A sequence diagram of the Markdown package typesetting a markdown document using the Lua command-line interface.

important disadvantages: The converted TEX documents are cached on the file system, taking up an increasing amount of space. Unless the TEX engine includes a Lua interpreter, the package also requires shell access, which opens the door for a malicious actor to access the system. Last, but not least, the complexity of the procedure also impedes debugging.

A solution to the above problems is to decouple the conversion from the typesetting. First, the user converts markdown documents to TEX. Then, she typesets the TEX documents using the `\input` TEX command (see Figure 2). Before the first step, the user can remove any previously cached TEX documents. Before the second step, she can transform the TEX documents according to her need. During the second step, she does not need to provide shell access to TEX. Since the individual steps are separated, the source of an error is immediately obvious.

To enable this workflow, version 2.6.0 of the Markdown package introduced the Lua CLI, which is a separate Lua program that can be executed from the shell. Create a text document named `example.md` with the following content:

```
Some of these words *are emphasized*.
Use two asterisks for **strong emphasis**.
```

Next, execute the `kpsewhich markdown-cli.lua` command to find the location of the Lua CLI, such as `/usr/local/texlive/2019/texmf-dist/scripts/markdown/markdown-cli.lua` on GNU/Linux with TEX Live 2019. Execute `texlua` ⟨*location of the Lua CLI*⟩ `-- example.md example.tex` to convert the `example.md` markdown document to TEX. Finally, create a text document named `document.tex` with the following content:

```
\documentclass{article}
\usepackage{markdown}
\begin{document}
\input example
\end{document}
```

Execute the `pdflatex document.tex` command to produce a document with one formatted paragraph.

2.3 Documenting LATEX packages

The `doc` LATEX package makes it possible to document a LATEX document by writing a second LATEX document in the comments. This approach to documentation is referred to as literate programming and is popular with LATEX package authors, as it keeps the documentation close to the documented code.

To encourage contributions and readability, documentation can benefit from the use of a lightweight markup language such as Markdown. To allow this use case, version 2.6.0 of the Markdown package introduced the `stripPercentSigns` option, which informs the Lua interpreter that it should strip percent signs from the beginnings of lines in a markdown document. Create a text document named `document.dtx` with the following content:

```
% \iffalse
\documentclass{ltxdoc}
\usepackage[stripPercentSigns]{markdown}
\begin{document}
\DocInput{document.dtx}
\end{document}
% \fi
% \begin{markdown}
% * Candy.
% * Gum.
% * Booze.
% \end{markdown}
```

Execute the following command to produce a document with an unordered list:

```
pdflatex -shell-escape document.dtx
```

2.4 Content slicing

Despite its simplicity (or perhaps because of it), Markdown has become a popular choice for writing all kinds of documents ranging from notes and lecture slides to books that span hundreds of pages. When typesetting these documents, it is often useful to typeset only a small part of them: Lecture slides spanning an entire term can be chopped into lectures. Bits and pieces from a ragtag of notes can be put together into a single coherent document. A behemoth of a book that takes thirty minutes to compile may take only one, when a single chapter is requested.

To make markdown documents more easily stylable, there exist syntax extensions for assigning HTML attributes to markdown elements. To enable typesetting only a part of a markdown document, version 2.7.0 of the Markdown package provides the `headerAttributes` and `slice` options. The `headerAttributes` option enables the Pandoc syntax for HTML attributes and the `slice` option specifies which part of a document should be typeset. Create a text document named `document.tex` with the following content:

```
\documentclass{article}
\usepackage[headerAttributes]{markdown}
\usepackage{filecontents}
\begin{filecontents*}{example.md}
# Palačinky
Crêpe-like pancakes, best served with jam.

## Step 3 {#step3}
Repeat step 2 until no batter is left.

## Step 1 {#step1}
Combine the ingredients and whisk until you
have a smooth batter.

## Step 2 {#step2}
Heat oil on a pan, pour in a tablespoonful of
the batter, and fry until golden brown.
\end{filecontents*}
\begin{document}
\markdownInput[slice=^ ^step3]{example.md}
\markdownInput[slice=step1 step2]{example.md}
\markdownInput[slice=step3]{example.md}
\end{document}
```

Execute the following command to produce a document with one section and three subsections:

```
pdflatex -shell-escape document.tex
```

3 Documentation

Presentation software, Tufte [6] argues, carries its own cognitive style that impedes communication. Similarly, literate programming tends to produce documentation that is poorly structured, because it adheres too closely to the documented code. Some LaTeX packages provide a user manual that is written independently of the documented code. While this improves readability, it also sacrifices literate programming and its ease of maintenance.

Before version 2.5.6, the Markdown package only provided technical documentation produced by literate programming. Since version 2.5.6, the package also provided a user manual aimed at the end user rather than a developer. This manual was, however, still produced using literate programming, leading to poor text structure. Since version 2.7.0, the user manual is combined from three texts describing user interfaces, package options, and markdown tokens, respectively. This leads to readable documentation without sacrificing literate programming.

4 Conclusion

TeX is a fine tool for typesetting many kinds of documents. It may, however, not be the best language for writing them. When preparing structurally simple documents, lightweight markup languages such as Markdown are often the best choice. In this article, I described the new features and documentation in version 2.7.0 of the Markdown package.

Acknowledgments

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

Support for content slicing was graciously sponsored by David Vins and Omedym.

References

[1] M. Dominici. An overview of Pandoc. *TUGboat* 35(1):44–50, 2014. `tug.org/TUGboat/tb35-1/tb109dominici.pdf`.

[2] J. Gruber. Daring Fireball: Markdown, 2013. `daringfireball.net/projects/markdown`.

[3] J. MacFarlane. Pandoc: A universal document converter, 2019. `pandoc.org`.

[4] F. Mittelbach. *The doc and shortvrb Packages*, 2018. `ctan.org/pkg/doc`.

[5] V. Novotný. Using Markdown inside TeX documents. *TUGboat* 38(2):214–217, 2014. `tug.org/TUGboat/tb38-2/tb119novotny.pdf`.

[6] E. R. Tufte. *The Cognitive Style of PowerPoint.* Graphics Press Cheshire, CT, 2nd edition, 2006.

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