Markdown Example

David Dervishi

2022-02-15

Contents

1 Introduction

Hi! This is an example Markdown document that can serve as an introduction to some common features. Be sure to check the Markdown source used to generate this document if you want to see a complete example!

1.1 Writing Markdown

You can use any text editor to write Markdown, and you need a Markdown compiler to transform your source into HTML, a PDF or any other format. There are many, *many* flavors of Markdown around providing possibly different features and using slightly different syntax. This document assumes you are using Pandoc[1], which you can find packaged in most GNU/Linux distributions. For example, this will install Pandoc on Debian-based system:

```
~# apt install pandoc
```

[1] https://pandoc.org/

Generating PDFs requires a LaTeX environment, which you can get by installing TeXLive for example:

```
~# apt install texlive-full
```

Once you have this all set up, you can write some Markdown in a file, say `example.md`, and compile it with Pandoc. The simplest way to do so is by using the following command:

```
~$ pandoc -s -o output_file example.md
```

Pandoc is usually smart enough to determine what kind of document you want by looking at the extension of the `output_file`. For example, if it ends with `.pdf`, it will produce a PDF, and an HTML file if it ends with `.html`. If you need to be specific, you can explicitly state the source and destination formats using

```
~$ pandoc -s --from markdown --to pdf -o output_file example.md
```

2 Basic Document structure

A simple document starts with a YAML block containing for example the title and authors of the document, followed by the actual document content.

2.1 Metadata Block

A metadata block looks like the following:

```
---
title: Your Nice title
author:
  - Your Name
  - Name Of Your Teammate
date: YYYY-MM-DD
---
```

Note that different authors are listed together, and that the block starts and ends with '`---`'. Many more options can be set in this block, and they can all be found in Pandoc's manual. You can find a couple of them in this document's source.

2.2 Content

Markdown is designed to be simple and easy to write and read, being mostly plain text with relatively little markup. But whatever document you may write, you usually end up structuring it by splitting it into different sections and paragraphs. Markdown lets you do this in a simple way:

```
# This is a level 1 header

This is text

## This is a level 2 header

This is text again

This is still text, but on a different paragraph because of the blank line above

### This is a level 3 header

You can keep adding headers, if you feel like it
```

3 Lists

Markdown lets you build both ordered and unordered lists.

3.1 Unordered Lists

You can create (possibly nested) lists with a sequence of bullets. The following code

```
* Item 1
* Item 2
  + Nested item 1
  + Nested item 2
* Item 3
```

Is rendered as

- Item 1
- Item 2
  - Nested item 1
  - Nested item 2
- Item 3

Which is simple enough.

3.2 Ordered Lists

Ordered lists are very similar to unordered lists, except that you have to define what kind of numbering will be used for the list elements. For example, you can write

```
1. Item 1
2. Item 2
   a) Item a
   b) Item b
3. Item 3
   i. Item i
   ii. Item ii
```

Which renders to

1. Item 1
2. Item 2
   a) Item a
   b) Item b
3. Item 3
   i. Item i
   ii. Item ii

But then you would have to do the whole numbering manually! How horrible!

You can alternatively let Pandoc do the work for you by letting it pick labels:

```
#. Item 1
#. Item 2
   #. Nested item 1
   #. Nested item 2
#. Item 3
```

1. Item 1
2. Item 2
    (a) Nested item 1
    (b) Nested item 2
3. Item 3

And if you want to pick a specific label, then you don't *really* have to do all the numbering: the first label in the list determines how all the other ones will be generated:

```
1. Item 1
#. Item 2
   i. Item i
   #. Item ii
# Item 3
```

1. Item 1
2. Item 2
    i. Item i
    ii. Item ii
3. Item 3

You should have observed by now that indentation in lists **does** matter.

4 Text Formatting

You can format text in several useful ways:

- You can `_emphasize things_` *you can also replace _ with **.*
- You can `__emphasize things a lot__` **you can also replace _ with ***
- You can `~~strike thing out~~` ~~like this~~
- You can `use^exponents^` like$^{\text{this}}$
- You can `use~indices~` like$_{\text{this}}$
- You can write inline code (with a typewriter font) like I did in the previous points with backticks ‘

5 Links And References

A link can be written as follows:

```
[placeholder](link)
```

For example GNUGeneration's website.

You can also use labels instead of direct links. For example, the following

```
This is a [link][gnugen]. If the label name is the same as your text, you may
omit the second pair of brackets: [gnugen].

[gnugen]: https://gnugeneration.epfl.ch
```

renders as

This is a link. If the label name is the same as your text, you may omit the second pair of brackets: gnugen.

You can then put your labels wherever you want them to be. Footnotes[2] have a very similar syntax:

[2] Like this one

```
This is a footnote[^label]

[^label]: Footnote text
```

You can also create references to your document using the format `[placeholder](#section-identifier)`, where the section identifier is automatically created from your header (see the Pandoc manual for details). For example, this section is identified by `links-and-references`, and here is a link to it.

6 Images

Images look like links with an exclamation mark. You can use links to files or remote content.

```
![caption](link)
```

Figure 1: Markdown logo

7 Code Blocks

You may find yourself wanting to embed code blocks inside your Markdown documents. This is very easy to do:

```
```<lang>
<code>
```
```

Blocks are delimited by three backticks. `lang` is a language identifier, like `markdown`.

You can also create blocks of raw code that are directly given to the relevant parser, and not interpreted as Markdown. This can be useful if you want to embed raw LaTeX or HTML or whatever code into your document without needing to escape anything.

```
```{=latex}
\begin{equation*}
    x_{1,2} = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}
\end{equation*}
```
```

$$x_{1,2} = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

8 Math and LaTeX

You can directly embed LaTeX commands inside your Markdown document, as long as you have a working environment. In particular, this means you can use inline math by writing whatever you want $between dollars$, like $ax^2 + bx + c = 0$. Other LaTeX environments are also supported if you need them. Here is an example with the `align` environment:

```
% observe that you don't necessarily have to write this as a raw code block as
% above
\begin{align*}
    x + 2 &= 0 \\
    x &= -2
\end{align*}
```

$$\begin{aligned} x + 2 &= 0 \\ x &= -2 \end{aligned}$$

9 Tables

Tables are easy to create in Markdown. To create a table, you simply have to draw a table! There are several ways to do that, but here are two examples:

```
| Product   | Cost | Quantity | Total |
|:---------|-----:|---------:|------:|
| Apple     | 5    | 4        | 20    |
| Broccoli | 2    | 8        | 16    |
| Chili     | 2    | 5        | 10    |
| Date      | 3    | 5        | 15    |
|           |      |          | 61    |

: Example 1

--------------------------------
Product   Cost  Quantity  Total
-------- ----- --------- ------
Apple        5         4     20

Broccoli     2         8     16

Chili        2         5     10

Date         3         5     15

                           61
--------------------------------

: Example 2
```

Table 1: Example 1

Product	Cost	Quantity	Total
Apple	5	4	20
Broccoli	2	8	16
Chili	2	5	10
Date	3	5	15
			61

Table 2: Example 2

Product	Cost	Quantity	Total
Apple	5	4	20
Broccoli	2	8	16
Chili	2	5	10
Date	3	5	15
			61

- Example 1: notice how the colon in the separation line between the header and contents defines the alignment of each column. One column on both sides gives you centering!
- Example 2: here, the alignment is also defined by the separation lines, but in a more subtle way. The "Product" title is aligned to the left of its line, i.e. there is one dash on the right of the title, and this means that the column is aligned to the left. The same holds for the other columns. One more dash on each side gives you centering!

In both cases, the table's caption can be specified on a line starting with a colon.

(And if you're anything like me, you'll end up writing your tables with embedded LaTeX instead)

```
\begin{table}[ht]
    \centering
    \caption{The same table in \LaTeX{}}
    \begin{tabular}{|l|r|r|r|}
        \hline Product  & Cost & Quantity & Total \\
        \hline Apple    & 5    & 4        & 20    \\
               Broccoli & 2    & 8        & 16    \\
               Chili    & 2    & 5        & 10    \\
               Date     & 3    & 5        & 15    \\
        \hline          &      &          & 61    \\
        \hline
    \end{tabular}
\end{table}
```

Table 3: The same table in LaTeX

Product	Cost	Quantity	Total
Apple	5	4	20
Broccoli	2	8	16
Chili	2	5	10
Date	3	5	15
			61