Markdown & Pandoc Essentials

arcasen

https://github.com/arcasen/markdown-include

2025 年 9 月 4 日

目录

第一章　简介

1.1　什么是 Markdown?

Markdown 是一种轻量级标记语言，你可以使用它向纯文本文档添加格式元素。Markdown 由 John Gruber 于 2004 年创建，现在是世界上最流行的标记语言之一。

使用 Markdown 与使用 WYSIWYG 编辑器不同。在 Microsoft Word 等应用程序中，你可以单击按钮来设置单词和短语的格式，并且更改会立即显示。Markdown 并非如此。当你创建一个 Markdown 格式的文件时，你向文本添加 Markdown 语法来指示哪些单词和短语应显示为不同格式。

例如，要表示标题，你可以在标题前添加一个井号（例如，`# 标题一`）。或者要使短语变为粗体，你可以在短语前后添加两个星号（例如，`**此文本为粗体**`）。在文本中看到 Markdown 语法可能需要一段时间才能适应，特别是如果你习惯于 WYSIWYG 应用程序。

你可以使用文本编辑器应用程序向纯文本文件添加 Markdown 格式元素。或者，你可以使用 macOS、Windows、Linux、iOS 和 Android 操作系统的众多 Markdown 应用程序之一。还有专门用于编写 Markdown 的几个基于 Web 的应用程序。

根据你使用的应用程序，你可能无法实时预览格式化的文档。但这没关系，Markdown 格式语法的主要设计目标是使其尽可能具有可读性。

1.2　为何使用 Markdown?

你可能想知道，人们为何使用 Markdown 而不是所见即所得编辑器。为什么在界面中按按钮格式化文本时还要使用 Markdown 来编写？事实证明，人们使用 Markdown 而不是所见即所得编辑器的原因有很多。

- Markdown 使语义（semantics）与格式（format）分离，让你更聚焦于语义。

- Markdown 是最简单的标记语言语言（markup languages）之一。
- Markdown 可用于一切。人们使用它来创建网站、文档、笔记、书籍、演示文稿、电子邮件和技术文档。
- Markdown 是可移植的。包含 Markdown 格式文本的文件几乎可以使用任何应用程序打开。如果你决定不喜欢当前使用的 Markdown 应用程序，你可以将 Markdown 文件导入另一个 Markdown 应用程序。这与 Microsoft Word 等将内容锁定为专有文件格式的文字处理应用程序形成了鲜明的对比。
- Markdown 与平台无关。你可以在运行任何操作系统的任何设备上创建 Markdown 格式的文本。
- Markdown 具有未来性。即使你使用的应用程序在未来某个时间点停止工作，你仍然可以使用文本编辑应用程序阅读 Markdown 格式的文本。对于需要无限期保存的书籍、大学论文和其他里程碑式文档，这是一个重要的考虑因素。
- Markdown 无处不在。像 Reddit 和 GitHub 这样的网站支持 Markdown，并且许多桌面和基于 Web 的应用程序也支持它。

1.3 Markdonw 是如何工作的?

当您使用 Markdown 书写时，文本会存储在具有.md 或.markdown 扩展名的纯文本文件中。但接下来呢? Markdown 格式的文件如何转换为 HTML 或可打印的文档?

简而言之，您需要一个能够处理 Markdown 文件的 Markdown 应用程序。

Markdown 应用程序使用称为 Markdown 处理器（通常也称为 “解析器” 或 “实现”）的东西，将 Markdown 格式的文本提取出来并将其输出为 HTML 格式。在这一点上，您的文档可以在网络浏览器中查看，或与样式表结合使用并打印出来。您可以在下面看到此过程的可视化表示。

Markdown 应用程序和处理器是两个独立的组件。为了简洁起见，我在下图中将它们合并为一个元素（“Markdown 应用程序”）。

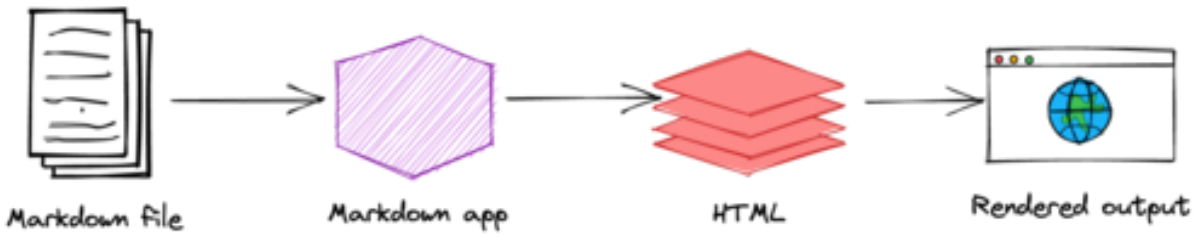

图 1.1: Markdown Flowchart

总而言之，这是一个由四部分组成的过程

1. 使用文本编辑器或专门的 Markdown 应用程序创建 Markdown 文件。该文件应具有.md 或.markdown 扩展名。
2. 在 Markdown 应用程序中打开 Markdown 文件。
3. 使用 Markdown 应用程序将 Markdown 文件转换为 HTML 文档。
4. 在网络浏览器中查看 HTML 文件，或使用 Markdown 应用程序将其转换为其他文件格式，例如 PDF。

1.4 Markdown 有什么用?

Markdown 是一种快速简便的方法，可用于做笔记、为网站创建内容以及生成可打印的文档。

学习 Markdown 语法并不需要很长时间，一旦您知道如何使用它，您就可以在几乎任何地方使用 Markdown 来书写。大多数人使用 Markdown 为网络创建内容，但 Markdown 适用于从电子邮件到购物清单的所有内容的格式化。

1.5 Markdown 的风格

使用 Markdown 最令人困惑的方面之一是几乎每个 Markdown 应用程序都实现了略有不同的 Markdown 版本。这些 Markdown 变体通常称为风格，如 GitHub Flavored Markdown，Pandoc Flavored Markdown 等等。

1.6 其他资源

有许多资源可用于学习 Markdown。以下是一些其他入门资源：

- John Gruber 的 Markdown 文档：由 Markdown 创建者编写的原始指南。
- Markdown 教程：一个开源网站，允许你在网络浏览器中尝试 Markdown。
- Awesome Markdown：Markdown 工具和学习资源列表。
- Markdown 排版：一个多部分系列，描述了使用 pandoc 和 ConTeXt 排版 Markdown 文档的生态系统。

第二章　Pandoc's Markdown

2.1　设计哲学 Philosophy

John Gruber 将 Markdown 设计为易于编写，并且更重要的是易于阅读：

> Markdown 格式的文档应该可以直接发布为纯文本，而不应看起来像是被标签或格式化指令标记过。

在 HTML 中，元素主要分为两类：块级元素（Block Elements）和内联元素（Span Elements，也称为行内元素）。这两类元素在页面布局和行为上有显著区别。

Markdown 基本语法包括：

- 块级元素（Block elements）：paragraphs、line breaks、headers、blockquotes、lists、code blocks、horizontal rules；
- 内联元素（Span elements）：links、emphasis、code、images；
- 其它元素（Miscellaneous）：backslash escape、automatic links。

Pandoc 中支持扩展修订版本的 Markdown 语法。Markdown 的设计哲学指导了 Pandoc 在寻找表格、脚注和其他扩展的语法时所做的决定。然而，Pandoc 的目标与 Markdown 最初的目标不同。虽然 Markdown 最初是为生成 HTML 而设计的，但 Pandoc 是为多种输出格式而设计的。因此，虽然 Pandoc 允许嵌入原始 HTML，但它并不鼓励这样做，并提供了其他非 HTML 方式来表示定义列表、表格、数学和脚注等重要文档元素。

- 使用 Pandoc 中支持的 Markdown 语法用 `-f markdown`
- 使用标准 Markdown 语法用 `-f markdown_strict`

Pandoc 所支持的语法各种对标准 Markdown 语法的扩展可以通过在格式后以 `+EXTENSION` 添加或`-EXTENSION` 去除，如：

- `-f markdown-footnotes` 表示识别除了 `footnotes` 扩展之外的所有 pandoc Markdown 语法

- `-f markdown_strict+footnotes+pipe_tables` 表示识别标准 Markdown 语法加上 `footnotes` 和 `pipe_tables` 扩展语法。

2.2 段落 Paragraphs

段落由一行或多行文本组成，后跟一行或多行空行。换行符被视为空格，因此您可以根据需要重新排列段落。

如果需要强制换行，请在行尾添加两个或更多空格。

示例：

```
<!-- 注意：下行末尾有两个空格，将导致换行 -->
Lorem ipsum dolor sit amet, consectetur
adipiscing elit.

Quisque consectetur, risus et suscipit dignissim.
```

Lorem ipsum dolor sit amet, consectetur
adipiscing elit.

Quisque consectetur, risus et suscipit dignissim.

2.2.1 扩展：`escaped_line_breaks`

反斜杠后跟换行符也是一种硬换行符。

在多行和网格表格单元格中,这是创建硬换行符的**唯一**方法,因为单元格中的尾随空格会被忽略，见扩展：*mutiline_tables* 和扩展：*grid_tables*。

示例：

```
<!-- 注意：反斜杠后跟换行符 (`\`)也是一种硬换行符 -->
```

```
Lorem ipsum dolor sit amet, consectetur\
adipiscing elit.

Quisque consectetur, risus et suscipit dignissim.
```

Lorem ipsum dolor sit amet, consectetur
adipiscing elit.

Quisque consectetur, risus et suscipit dignissim.

2.3 标题 Headings

Pandoc 可以使用两种标题格式：Setext 和 ATX。

2.3.1 Setext 样式

Setext 样式的标题是一行带有 “下划线” 的文本，其中带有一行 = 符号（对于一级标题）或 -符号（对于二级标题）。标题文本可能包括内联格式 *Inline Formatting*，例如强调和斜体。

示例：

```
A level-one heading
===================

A level-two heading
-------------------

A *formatted* heading
=====================

Introduction to [Makedown Guide](https://www.markdownguide.com)
---------------------------
```

2.3.2 ATX 格式

ATX 样式标题由 1 到 6 个连续的#符号和一行文本组成，在行尾可能有任意数量的符号。行首的符号#数量即为标题的级别。与 Setext 标题一样，ATX 标题允许内联格式 *Inline Formatting*。

示例：

```
# A level-one heading

## A level-two heading

### A *formatted* heading

#### Introduction to [Makedown Guide](https://www.markdownguide.com)
```

2.3.3 扩展：blank_before_header

原始 Markdown 语法不需要标题前有空行。Pandoc 确实需要这个（当然，文档的开头除外）。提出这一要求的原因是，# 很容易意外地出现在一行的开头（可能由于换行）。

示例：

```
I like several of their flavors of ice cream:
# 22, for example, and # 5.
```

2.3.4 扩展：space_in_atx_header

许多 Markdown 实现并不要求 ATX 标题开头的 # 与标题文本之间有空格，因此 `# heading 1` 和 `#heading 1` 都算作标题。

Pandoc 默认要求 # 与标题文本之间有空格。如果要取消这个要求，编译时使用选项 `-f markdown-space_in_atx_header`。

示例：

```
#head

# head
```

编译时执行命令：`pandoc -f markdown-space_in_atx_header input.md -o output.md`。

2.3.5 扩展：auto_identifiers

没有明确指定标识符的标题将根据标题文本自动分配唯一标识符。

从标题文本中获取标识符的默认算法是：

- 删除所有格式、链接等。
- 删除所有脚注。
- 删除所有非字母数字字符，下划线、连字符和句点除外。
- 用连字符替换所有空格和换行符。
- 将所有字母字符转换为小写。
- 删除第一个字母之前的所有内容（标识符不能以数字或标点符号开头）。
- 如果此后没有剩余内容，则使用标识符 `section`。

例如：

标题	标识符
Heading identifiers in HTML	heading-identifiers-in-html
Maître d'hôtel	maître-dhôtel
Dogs?–in *my* house?	dogs–in-my-house
[HTML], [S5], or [RTF]?	html-s5-or-rtf
3. Applications	applications
33	section

在大多数情况下，这些规则应该允许根据标题文本确定标识符。例外情况是多个标题具有相同的文本；在这种情况下，第一个标题将获得如上所述的标识符；第二个标题将获得相同的标识符并附加 `-1`；第三个标题将附加 `-2`；依此类推。

但是，如果启用 `gfm_auto_identifiers`，则会使用不同的算法；请参见下文。

这些标识符用于在选项生成的目录中提供链接目标 `--toc|--table-of-contents`。它们还可以轻松地提供从文档某个部分到另一个部分的链接。例如，指向此部分的链接可能如下所示：

```
See the section on
[heading identifiers](#heading-identifiers-in-html-latex-and-context
   ).
```

但请注意，这种提供章节链接的方法仅适用于 HTML、LaTeX 和 ConTeXt 格式。

如果 `--section-divs` 指定了该选项，则每个部分将被包裹在 section（或 div，如果指定了 html4）中，并且标识符将附加到封闭的 `<section>`（或 `<div>`）标签而不是标题本身。这允许使用 JavaScript 操作整个部分，或在 CSS 中进行不同的处理。

2.3.6 扩展：ascii_identifiers

使生成的标识符为 `auto_identifiers` 纯 ASCII 码。带重音符号的拉丁字母中的重音符号会被去除，非拉丁字母会被省略。

2.3.7 扩展：gfm_auto_identifiers

更改 `auto_identifiers` 使用的算法以符合 GitHub 的方法。空格将转换为短划线 (-)，大写字母将转换为小写字母，除-和 _ 之外的标点符号将被删除。表情符号将替换为其名称。

2.3.8 扩展：header_attributes

可以在包含标题文本的行末尾使用以下语法为标题分配属性：

```
{#identifier .class .class key=value key=value}
```

请注意，虽然此语法允许分配类和键/值属性，但编写者通常不会使用所有这些信息。标识符、类和键/值属性用于 HTML 和基于 HTML 的格式（例如 EPUB 和 slidy）。标识符用于 LaTeX、ConTeXt、Textile、Jira 标记和 AsciiDoc 编写器中的标签和链接锚点。

如果指定了带有 `unnumbered` 类的标题，则即使使用 `--number-sections` 也不会被编号。属性上下文中的单个连字符（-）相当于 `.unnumbered`，并且在非英语文档中更可取。因此，

```
# My heading {-}
```

和

```
# My heading {.unnumbered}
```

如果存在 unnumbered 和 unlisted，则标题将不会包含在目录中。（目前此功能仅适用于某些格式：基于 LaTeX 和 HTML、PowerPoint 和 RTF 的格式。）

不编号的标题仍然会出现在目录中，除非你明确排除它们（例如，使用 {.unlisted} 属性）。

示例：

```
# Heading identifiers in HTML {#foo}

你可以这样使用：

[标题 1](#foo)


# 中文标题 identifiers in HTML {#bar}

你可以这样使用：

[标题 2](#bar)
```

2.3.9 扩展：implicit_header_references

Pandoc 的行为就像每个标题都定义了引用链接（reference links）一样。因此，要链接到标题，只需要直接在标题加上[]，并且可以使用内联格式 *Inline Formatting*。

如要链接到标题：

```
# Heading identifiers in HTML
```

你可以简单地写：

```
[Heading identifiers in HTML]
```

或者（使用斜体）

```
*[Heading identifiers in HTML]*
```

或者

```
[Heading identifiers in HTML][]
```

或者

```
[the section on heading identifiers][heading identifiers in
HTML]
```

而不是明确给出标识符：

```
[Heading identifiers in HTML](#heading-identifiers-in-html)
```

如果有多个标题具有相同的文本，则相应的参考将仅链接到第一个标题，并且您需要使用明确的链接来链接到其他标题，如上所述。

与常规参考链接一样，这些参考不区分大小写。

显式链接引用定义始终优先于隐式标题引用。因此，在下面的例子中，链接将指向 bar，而不是 #foo：

```
# Foo

[foo]: bar

See [foo]
```

下面是一个完整示例：

```
# Heading identifiers in HTML

你可以简单地写

[Heading identifiers in HTML]

或者

[Heading identifiers in HTML][]

或者

[the section on heading identifiers][heading identifiers in
HTML]

而不是明确给出标识符：

[Heading identifiers in HTML](#heading-identifiers-in-html)


# 中文标题 identifiers in HTML
```

```
你可以简单地写

[中文标题 identifiers in HTML]

或者

[中文标题 identifiers in HTML][]

或者

[the section on heading identifiers][中文标题 identifiers in
HTML]

而不是明确给出标识符:

[中文标题 identifiers in HTML](#中文标题-identifiers-in-html)
```

2.4 引用块 Blockquotes

Markdown 使用电子邮件约定来引用文本块。块引用是指一个或多个段落或其他块元素（例如列表或标题），每行前面都有一个 > 字符和一个可选的空格。（引用 > 不必从左边距开始，但缩进不应超过三个空格。）

示例:

```
> This is a block quote. This
> paragraph has two lines.
>
> 1. This is a list inside a block quote.
> 2. Second item.
```

This is a block quote. This paragraph has two lines.

1. This is a list inside a block quote.
2. Second item.

还允许使用 “懒惰” 形式，即 > 仅要求在每个块的第一行出现字符:

```
> This is a block quote. This
paragraph has two lines.
```

```
> 1. This is a list inside a block quote.
2. Second item.
```

> This is a block quote. This paragraph has two lines.
>
> 1. This is a list inside a block quote.
> 2. Second item.

块引用中可以包含其他块引用的块元素。也就是说，块引用可以嵌套：

```
> This is a block quote.
>
> > A block quote within a block quote.
```

> This is a block quote.
>
> > A block quote within a block quote.

如果该 > 字符后跟一个可选空格，则该空格将被视为块引用标记的一部分，而不是内容缩进的一部分。因此，要将缩进的代码块放入块引用中，你需要在 > 后添加五个空格：

```
含有代码的引用：

>     print('hello,world')
```

> 含有代码的引用：
>
> ```
> print('hello,world')
> ```

2.4.1 扩展：blank_before_blockquote

原始 Markdown 语法并不要求块引用前有空行。Pandoc 则要求这样做（当然，文档开头除外）。这样做的原因是，> 很容易意外地出现在行首（例如由于换行）。因此，除非 markdown_strict 使用这种格式，否则以下代码在 Pandoc 中不会生成嵌套的块引用：

示例：

```
This is NOT a block quote.
> Lorem ipsum dolor sit amet, consectetur adipiscing elit.
```

This is NOT a block quote. > Lorem ipsum dolor sit amet, consectetur adipiscing elit.

示例：

```
> This is a block quote.
>> Not nested, since `blank_before_blockquote` is enabled by default
```

> This is a block quote. > Not nested, since blank_before_blockquote is enabled by default

---

上面的嵌套块引用中需要加入空行（不加入空行则不会变成块引用）。

示例：

```
block qoutes

> This is a block quote.
>
> > A block quote within a block quote.
```

---

block qoutes

> This is a block quote.
>
> > A block quote within a block quote.

2.5 代码块 Code Blocks

Pandoc 支持两种代码块：

- 缩进式代码块（indented code blocks，Markdown 基本语法）
- 围栏式代码块（fenced code blocks）

2.5.1 缩进式代码块

缩进四个空格（或一个制表符）的文本块将被视为逐字文本：也就是说，特殊字符不会触发特殊格式，所有空格和换行符都会保留。例如：

```
示例代码：<!-- 代码块必须用空行与周围文本分隔开 -->

    if (a > 3) {
      moveShip(5 * gravity, DOWN);
    }
```

示例代码：

```
if (a > 3) {
  moveShip(5 * gravity, DOWN);
}
```

初始（四个空格或一个制表符）缩进不被视为逐字文本的一部分，并将在输出中删除。

逐字文本中的空白行不必以四个空格开头。

2.5.2 扩展：fenced_code_blocks

除了标准缩进式代码块外，Pandoc 还支持带围栏式代码块。这些代码块以一行三个或更多波浪号 (~) 开始，以一行波浪号结束，波浪号的长度必须至少与起始行一样长。这些行之间的所有内容都被视为代码。无需缩进：

```
~~~~~~~
if (a > 3) {
  moveShip(5 * gravity, DOWN);
}
~~~~~~~
```

与常规代码块一样，围栏式代码块必须用空行与周围文本分隔开。

如果代码本身包含一行波浪号或反引号，则只需在开始和结束处使用一排较长的波浪号或反引号：

```
~~~~~~~~~~
code including tildes
~~~~~~~~~~
```

2.5.3 扩展：backtick_code_blocks

与 fenced_code_blocks 相同，但使用反引号 (‘) 而不是波浪号 (~)。

2.5.4 扩展：fenced_code_attributes

或者，您可以使用以下语法将属性附加到隔离或反引号代码块：

```
```{#helloworld .c .numberLines startFrom="101" .lineAnchors caption
   ="Hello, World"}
#include <stdio.h>

int main(int argc, const char * argv[]) {
  printf("Hello, World!\n");
  return 0;
}
```
```

代码 2.1: Hello, World

```c
#include <stdio.h>

int main(int argc, const char * argv[]) {
  printf("Hello, World!\n");
  return 0;
}
```

这里 `#helloworld` 是一个标识符，可以使用 `[Helloworld](#helloworld)` 建立链接；`c` 和 `numberLines` 是类，并且 `startFrom` 是一个值为 101 的属性。某些输出格式可以使用此信息进行语法高亮显示。目前，唯一使用此信息的输出格式是 HTML、LaTeX、Docx、Ms 和 PowerPoint。如果您的输出格式和语言支持高亮显示，则上面的代码块将高亮显示，并带有编号行。（要查看支持哪些语言，请输入 `pandoc --list-highlight-languages`。）

`numberLines`（或 `number-lines`）类将使代码块的行以 1 或 `startFrom` 的属性值开始进行编号。`lineAnchors`（或 `line-anchors`）类将使这些行在 HTML 输出中成为可点击的锚点。

2.5.5 语法高亮显示

Pandoc 会在标记了语言名称的代码块中自动高亮语法，使用 Haskell 库 skylighting 进行高亮。目前仅支持 HTML、EPUB、Docx、Ms、Man 和 LaTeX/PDF 输出的高亮功能。要查看 pandoc 识别的语言名称列表，请输入 `pandoc --list-highlight-languages`。

语法高亮的风格通过样式表的变化进行控制：HTML 输出使用级联样式表（CSS），LaTeX（以及 PDF）输出使用一组 `\newcommand` 选项。这些高亮指令直接嵌入输出文件中（当生成独立文档时），因此不易覆盖。不过，你可以从预定义的高亮样式列表中选择。运行以下命令可查看完整样式列表：

```
pandoc --list-highlight-styles
```

默认配色方案是 `pygments`，模仿了 Python 库 pygments 的默认配色（尽管实际高亮并未使用 pygments）。使用 `--highlight-style` 选项选择样式。例如，我在构建书籍的打印版本时想要黑白输出，因此使用选项：

```
pandoc --highlight-style=monochrome
```

这会生成黑白高亮，使用斜体和粗体来显示不同的语言组件。你也可以在使用代码

块的同时完全禁用语法高亮，只需使用选项 `--no-highlight`。

如果对预定义样式不满意，可以使用 `--print-highlight-style` 生成一个 JSON 格式的 `.theme` 文件，修改后可作为 `--highlight-style` 的参数。例如，获取 pygments 样式的 JSON 版本：

```
pandoc -o my.theme --print-highlight-style pygments
```

然后编辑 `my.theme`，并按以下方式使用：

```
pandoc --highlight-style my.theme
```

如果内置高亮不满意，或想高亮不支持的语言，可以使用 `--syntax-definition` 选项加载 KDE 风格的 XML 语法定义文件。在编写自己的语法定义前，可参考 KDE 的语法定义仓库。

如果遇到 pandoc 报错 “无法读取高亮主题”，请检查 JSON 文件是否使用 UTF-8 编码且没有字节顺序标记 (BOM)。

输出格式为 LaTeX 或 PDF 时，

- 没有设置代码属性时，默认使用 `verbatim` 宏；
- 设置代码属性时，使用 Pandoc 自定义的 `Shaded` 宏[1]；
- 在编译时采用选项 `--listings`（也可以在 YAML 中设置 `listings: true`），则输出文件使用 `listings` 宏。

2.5.6 非断行空格 Non-breaking Space

在使用 Pandoc 将 Markdown 转换为其他格式（如 HTML 或 PDF）时，含有前导空格的代码（如 `  hello,world`）可能会忽略前导空格，即渲染为：`hello,world`。

在 Markdown 中，可以手动将前导空格替换为 Unicode 非断行空格（Non-breaking Space，简称 NBSP）。非断行空格可以通过文本编辑器或脚本替换输入，也可以直接输入：

- Windows：`Alt + 0160`（小键盘）
- Mac：`Option + Space`
- HTML/XML：` `
- LaTeX：`~`（波浪号）

[1]此时 `--highlight-style` 和 `--no-highlight` 选项才起作用，使用 `--listings` 时，格式由 `listings` 宏设置。

- Unicode：`U+00A0`

非断行空格的主要用途：

1. 防止自动换行。普通空格（U+0020）在文本换行时会被断开，但 NBSP 会强制让相连的单词或字符保持在同一行，避免被分开。示例：

```
100 km/h  <!-- "100 km/h" 会始终显示在同一行 -->
```

2. 在 HTML 中保留空格。HTML 默认会合并多个普通空格为一个，使用 ` ` 可以保留连续的空格。示例：

```
你好   世界  <!-- 显示为 "你好    世界"（3个空格） -->
```

3. 用于对齐或固定格式。在表格、代码等需要对齐的场景下，可以用 NBSP 代替普通空格，确保格式稳定。

- 过度使用 ` ` 可能导致代码可读性下降，建议在真正需要时使用。
- 在响应式设计中，依赖 NBSP 控制布局可能不灵活，推荐使用 CSS（如 `white-space: nowrap`）替代。

2.6 行块 Line Blocks

2.6.1 扩展：`line_blocks`

行块是由一系列以竖线 (|) 开头、后跟空格的行组成的序列。行的划分以及前导空格将在输出中保留；否则，这些行将被格式化为 Markdown 格式。这对于诗句和地址很有用：

```
| The limerick packs laughs anatomical
| In space that is quite economical.
|    But the good ones I've seen
|    So seldom are clean
| And the clean ones so seldom are comical

| 200 Main St.
| Berkeley, CA 94718
```

The limerick packs laughs anatomical
In space that is quite economical.
But the good ones I've seen
So seldom are clean
And the clean ones so seldom are comical

200 Main St.
Berkeley, CA 94718

如果需要，可以将行硬换行，但续行必须以空格开头：

```
| The Right Honorable Most Venerable and Righteous Samuel L.
  Constable, Jr.
| 200 Main St.
| Berkeley, CA 94718
```

The Right Honorable Most Venerable and Righteous Samuel L. Constable, Jr.
200 Main St.
Berkeley, CA 94718

内容中允许使用行内格式（例如强调）（但不能跨越行边界）。块级格式（例如块引用或列表）无法识别。

此语法借用自 reStructuredText。

2.7 列表 Lists

2.7.1 无序列表

项目符号列表是由项目符号列表项组成的列表。项目符号列表项以项目符号（*、+ 或 -）开头，列表前必须添加空行。如：

```
<!-- 列表前必须添加空行 -->
“紧凑”的列表：

* one
* two
```

```
* three

“松散”的列表：

* one

* two

* three

列表各行对齐看起来更美观：

* here is my first
  list item.
* and my second.

但是 Markdown 允许“懒惰”的格式（不需要对齐）：

* here is my first
list item.
* and my second.
```

“紧凑”的列表：

- one
- two
- three

“松散”的列表：

- one
- two
- three

列表各行对齐看起来更美观：

- here is my first list item.
- and my second.

但是 Markdown 允许“懒惰”的格式（不需要对齐）：

- here is my first list item.
- and my second.

列表项可以包含其他列表。在这种情况下，前面的空行是可选的。嵌套列表必须缩进，以与包含列表项的列表标记后的第一个非空格字符对齐。

2.7.2 列表项含有多个段落

列表项可以包含多个段落和其他块级内容。然而，后续段落必须以一个空行开头，并且缩进以与列表标记后的第一个非空格内容对齐。

例外情况：如果列表标记后跟一个缩进的代码块，该代码块必须在列表标记后 5 个空格开始，那么后续段落必须在列表标记最后一个字符后两个空格处开始。

```
* First paragraph.

  Continued.

* Second paragraph. With a code block, which must be indented with
   spaces:

       print('hello,world')
* `print('hello,world')`
```

- First paragraph.

  Continued.

- Second paragraph. With a code block, which must be indented with spaces:

  ```
  print('hello,world')
  ```

- `print('hello,world')`

2.7.3 嵌套列表

列表项可以包含其他列表。在这种情况下，前面的空行是可选的。嵌套列表必须缩进，以与包含它的列表项的列表标记后的第一个非空格字符对齐。

```
* fruits
  + apples
```

```
    - macintosh
    - red delicious
  + pears
  + peaches
* vegetables
  + broccoli
  + chard
```

- fruits
    - apples
        - macintosh
        - red delicious
    - pears
    - peaches
- vegetables
    - broccoli
    - chard

在 LaTeX 中，列表（包括 itemize、enumerate 和 description 环境）默认支持四级嵌套；否则，LaTeX 会报错，提示 `Too deeply nested`。在 HTML 中，列表（包括无序列表 `<ul>` 和有序列表 `<ol>`）的嵌套层级没有明确的上限，理论上可以无限嵌套。

2.7.4 有序列表

有序列表的工作方式与项目符号列表相同，只是项目以枚举器而不是项目符号开头。

在原始 Markdown 中，枚举器是十进制数字，后跟一个句点和一个空格。数字本身会被忽略，如：

```
有序列表 1:

1.  one
2.  two
3.  three
```

```
有序列表 2:

7.  one
5.  two
6.  three
```

有序列表 1:

1. one
2. two
3. three

有序列表 2:

7. one
8. two
9. three

2.7.5 扩展：fancy_lists

与原始 Markdown 不同，Pandoc 允许使用大小写字母和罗马数字以及阿拉伯数字来标记有序列表项。列表标记可以用括号括起来，也可以后跟一个右括号或句点。它们必须与后面的文本至少间隔一个空格；如果列表标记是带句点的大写字母，则必须至少间隔两个空格。

该 `fancy_lists` 扩展还允许使用 `#` 代替数字作为有序列表标记：

```
fancy lists 1:

a. one
#. two

fancy lists 2:

b) one
#) two

fancy lists 3:

A.  one
#.  two
```

```
fancy lists 4:

#. one
#. two
```

---

fancy lists 1:

a. one
b. two

fancy lists 2:

b) one
c) two

fancy lists 3:

A. one
B. two

fancy lists 4:

1. one
2. two

2.7.6 扩展：task_lists

Pandoc 支持任务列表，使用 GitHub Flavored Markdown 的语法。

```
任务列表示例：

- [ ] an unchecked task list item
- [x] checked item
```

任务列表示例：

☐ an unchecked task list item
☒ checked item

2.7.7 扩展：definition_lists

Pandoc 支持定义列表，使用 PHP Markdown Extra 的语法，并带有一些扩展。

```
Term 1

:   Definition 1

Term 2 with *inline markup*

:   Definition 2

        { some code, part of Definition 2 }

    Third paragraph of definition 2.
```

Term 1 Definition 1

Term 2 with *inline markup* Definition 2

```
{ some code, part of Definition 2 }
```

Third paragraph of definition 2.

每个术语必须占一行，行尾可以空一行，并且必须跟一个或多个定义。定义以冒号或波浪号开头，可以缩进一至两个空格。

一个术语可能有多个定义，每个定义可以由一个或多个块元素（段落、代码块、列表等）组成，每个块元素缩进四个空格或一个制表位。定义的主体（不包括第一行）应该缩进四个空格。但是，与其他 Markdown 列表一样，除了段落或其他块元素的开头外，你可以 “懒惰地” 省略缩进。

```
Term 1

:   Definition
with lazy continuation.

    Second paragraph of the definition.
```

Term 1 Definition with lazy continuation.

Second paragraph of the definition.

如果你在定义前留有空格（如上例所示），定义的文本将被视为段落。在某些输出格式中，这意味着术语/定义对之间的间距会更大。要获得更紧凑的定义列表，请省略定义前的空格：

```
Term 1
  ~ Definition 1

Term 2
  ~ Definition 2a
  ~ Definition 2b
```

Term 1 Definition 1

Term 2 Definition 2a

Definition 2b

请注意，定义列表中的项目之间需要有空格。

2.7.8 列表的结束

如果您想在列表后放置缩进的代码块怎么办？要“截断”第二项之后的列表，您可以插入一些非缩进的内容，例如 HTML 注释，这样不会以任何格式产生可见的输出；如果您想要两个连续的列表而不是一个大列表，则可以使用相同的技巧：

```
-   item one
-   item two

<!-- end of list -->

    { my code block }


1.  one
2.  two
```

```
3.  three

<!-- end of list -->

1.  uno
5.  dos
6.  tres
```

- item one
- item two

```
{ my code block }
```

1. one
2. two
3. three

1. uno
2. dos
3. tres

2.7.9 紧凑型列表

在使用 Pandoc 将 Markdown 列表转换为 LaTeX 时，是否使用 `\tightlist` 取决于 Pandoc 的版本、列表的上下文以及输出的 LaTeX 模板。如果列表嵌套在其他环境中（例如表格、定义列表等）或列表项间存在空行，Pandoc 可能不会插入 `\tightlist`。

2.8 分隔线 Horizontal Rules

要创建分隔线，请在单独一行上使用三个或多个星号 (`***`)、破折号 (`---`) 或下划线 (`___`)，并且不能包含其他内容。

示例：

```
***

---
```

```
_____________________
```

以上三个分隔线的渲染效果看起来都一样：

我们强烈建议水平线与周围文本之间使用空行分隔。如果水平线后没有空行，pandoc 可能会尝试将其后的行解释为 YAML 元数据块或表格。

2.9 表格 Tables

Pandoc 中支持四种列表样式，前三种要求等宽字体（fixed-width font）[2]，最后一种可以使用比例间隔字体（proportionally spaced font）[3]：

- `simple_tables`
- `multiline_tables`
- `grid_tables`
- `pipe_tables`

中文排版时该对齐是不好控制的，因为中文字体不属于等宽字体。

2.9.1 扩展：`table_captions`

所有 4 种表格类型均可选择添加标题（如下例所示）。标题是一个以字符串 `Table:`（或 `table:` 或仅 `:`）开头的段落，可以出现在表格之前或之后。

[2]Fixed-width Font（等宽字体），又称为 Monospaced Font，是一种所有字符（包括字母、数字、标点符号等）占据相同水平宽度的字体。

[3]Proportionally Spaced Font（比例间隔字体）是一种字符宽度不固定的字体，每个字符根据其实际形状占据不同的水平空间。现代中文字体（如思源黑体、苹方、微软雅黑）多为比例间隔字体，尤其在标点和中西文混排时体现宽度优化，适合出版、网页、UI 设计等。

- 当在表格之前使用 `:` `<table caption>` 形式的标题时，为避免解析成定义列表（见扩展：*definition_lists*），可在前面增加一空白行，或将其置于表格之后。
- Pandoc 的原生 Markdown 语法不支持在表格标题中直接添加脚注（见脚注 *Footnotes*）。当你在表格标题中使用脚注语法 `[^1]` 并尝试生成文件时，Pandoc 通常会将其作为纯文本处理，而不会将其渲染为正确的脚注格式。

2.9.2 扩展：simple_tables

表头和表行必须各占一行。列对齐由表头文本相对于其下方虚线的位置决定：

- 如果虚线与右侧的标题文本齐平，但左侧超出标题文本，则该列右对齐。
- 如果虚线与左侧的标题文本齐平，但在右侧超出标题文本，则该列左对齐。
- 如果虚线超出两侧的标题文本，则该列居中。
- 如果虚线与两侧的标题文本齐平，则使用默认对齐方式（在大多数情况下，将保持对齐）。

表格必须以一个空白行结束，或者以一行 dashes 后跟一个空白行结束。

可以省略列标题行，但表格结尾必须使用虚线。当省略标题行时，列对齐将根据表主体的第一行确定。

示例：

```
<!-- simple table -->

Table: 简单表格示例

  Right     Left     Center     Default
-------     ------ ----------   -------
     12     12        12            12
    123     123       123          123
      1     1          1             1

Table: 省略列标题行的简单列表示例

-------     ------ ----------   -------
     12     12        12            12
    123     123       123          123
      1     1          1             1
-------     ------ ----------   -------
```

表 2.2: 简单表格示例

Right	Left	Center	Default
12	12	12	12
123	123	123	123
1	1	1	1

表 2.3: 省略列标题行的简单列表示例

12	12	12	12
123	123	123	123
1	1	1	1

2.9.3 扩展：mutiline_tables

多行表格允许标题行和表格行跨越多行文本（但不支持跨越表格多列或多行的单元格）。

多行表格的工作方式类似于简单表格，但有以下区别：

- 必须以一行 dashes 开头，位于标题文本之前（除非省略标题行）。
- 必须以一排 dashes 结尾，然后是一个空行。
- 行与行之间必须用空行分隔。

在多行表格中，表格解析器会关注列的宽度，编写器会尝试在输出中重现这些相对宽度。因此，如果您发现输出中某一列太窄，请尝试在 Markdown 源代码中将其加宽。

在多行表格可以省略表头。

多行表可能只有一行，但该行后面应该跟着一个空行（然后是结束表格的 dashes 行），否则该表可能会被解释为简单表格。

示例：

```
-------------------------------------------------------------
 Centered   Default           Right Left
  Header    Aligned         Aligned Aligned
```

```
----------- ------- --------------- -------------------------
   First    row                12.0 Example of a row that
                                    spans multiple lines.

  Second    row                 5.0 Here's another one. Note
                                    the blank line between
                                    rows.
----------------------------------------------------------------
Table: Here's the caption. It, too, may span
multiple lines.

----------- ------- --------------- -------------------------
   First    row                12.0 Example of a row that
                                    spans multiple lines.

  Second    row                 5.0 Here's another one. Note
                                    the blank line between
                                    rows.
----------- ------- --------------- -------------------------

: Here's a multiline table without a header.

----------- ------- --------------- -------------------------
   First    row                12.0 Example of a row that
                                    spans multiple lines.

----------- ------- --------------- -------------------------

: Here's a multiline table with just one row.
```

表 2.4: Here's the caption. It, too, may span multiple lines.

Centered Header	Default Aligned	Right Aligned	Left Aligned
First	row	12.0	Example of a row that spans multiple lines.
Second	row	5.0	Here's another one. Note the blank line between rows.

表 2.5: Here's a multiline table without a header.

First	row	12.0	Example of a row that spans multiple lines.
Second	row	5.0	Here's another one. Note the blank line between rows.

表 2.6: Here's a multiline table with just one row.

First	row	12.0	Example of a row that spans multiple lines.

2.9.4 扩展：grid_tables

= 行将表头与表体分隔开来，对于无表头的表格，可以省略该行。网格表格的单元格可以包含任意块元素（多个段落、代码块、列表等）。

单元格可以跨越多列或多行[4]。

表头可能包含多行。

可以像管道表格一样指定对齐方式，通过在标题后的分隔线边界处放置冒号 :。

对于无表头的表，冒号位于顶行。

可以通过使用分隔线（=而不是-）来定义表格脚注。

表格脚注必须始终放在表格的最底部。

可以使用 Emacs 的表格模式 (M-x table-insert) 轻松创建网格表。

示例：

```
: 简单的网格表格

+---------------+---------------+--------------------+
| Fruit         | Price         | Advantages         |
+===============+===============+====================+
| Bananas       | $1.34         | - built-in wrapper |
```

[4]对于含有中文的表格，可能会失效。

```
|                |                | - bright color     |
+----------------+----------------+--------------------+
| Oranges        | $2.10          | - cures scurvy     |
|                |                | - tasty            |
+----------------+----------------+--------------------+
```

: 单元格可以跨越多列或多行

```
+---------------------+----------+
| Property            | Earth    |
+=============+=======+==========+
|             | min   | -89.2 °C |
| Temperature +-------+----------+
| 1961-1990   | mean  | 14 °C    |
|             +-------+----------+
|             | max   | 56.7 °C  |
+-------------+-------+----------+
```

: 表头可能包含多行

```
+---------------------+-----------------------+
| Location            | Temperature 1961-1990 |
|                     | in degree Celsius     |
|                     +-------+-------+-------+
|                     | min   | mean  | max   |
+=====================+=======+=======+=======+
| Antarctica          | -89.2 | N/A   | 19.8  |
+---------------------+-------+-------+-------+
| Earth               | -89.2 | 14    | 56.7  |
+---------------------+-------+-------+-------+
```

: 单元格对齐方式

```
+---------------+---------------+--------------------+
| Right         | Left          | Centered           |
+==============:+:==============+:==================:+
| Bananas       | $1.34         | built-in wrapper   |
+---------------+---------------+--------------------+
```

: 无表头的表格对齐

```
+--------------:+:--------------+:------------------:+
| Right         | Left          | Centered           |
+---------------+---------------+--------------------+
```

: 表格脚注

```
+---------------+---------------+
```

```
| Fruit          | Price          |
+================+================+
| Bananas        | $1.34          |
+----------------+----------------+
| Oranges        | $2.10          |
+================+================+
| Sum            | $3.44          |
+================+================+

<!-- 注意：多行和网格表格单元格只能用反斜杠换行 -->

: 多行和网格表格单元格中使用反斜杠换行

+----------------+----------------+--------------------+
| Fruit          | Price          | Advantages         |
+================+================+====================+
| Bananas        | first line\    | first line\        |
|                | next line      | next line          |
+----------------+----------------+--------------------+
| Bananas        | first line\    | first line\        |
|                | next line      | next line          |
+----------------+----------------+--------------------+
```

表 2.7: 简单的网格表格

Fruit	Price	Advantages
Bananas	$1.34	• built-in wrapper • bright color
Oranges	$2.10	• cures scurvy • tasty

表 2.8: 单元格可以跨越多列或多行

Property		Earth
Temperature 1961-1990	min	-89.2 °C
	mean	14 °C
	max	56.7 °C

表 2.9: 表头可能包含多行

Location	Temperature 1961-1990 in degree Celsius		
	min	mean	max
Antarctica	-89.2	N/A	19.8
Earth	-89.2	14	56.7

表 2.10: 单元格对齐方式

Right	Left	Centered
Bananas	$1.34	built-in wrapper

表 2.11: 无表头的表格对齐

Right	Left	Centered

表 2.12: 表格脚注

Fruit	Price
Bananas	$1.34
Oranges	$2.10
Sum	$3.44

表 2.13: 多行和网格表格单元格中使用反斜杠换行

Fruit	Price	Advantages
Bananas	first line	first line
	next line	next line
Bananas	first line	first line
	next line	next line

2.9.5 扩展：pipe_tables

管道表格语法与 PHP Markdown Extra 表格 表相同。起始和结束竖线字符（|）是可选的，但所有列之间都需要竖线。冒号指示列对齐方式。表头不可省略。要模拟无表头表格，请添加一个带有空白单元格的表头。由于管道指示列边界，因此列无需像上例那样垂直对齐。

管道表格的单元格不能包含段落和列表等块元素，也不能跨越多行。如果 Markdown 源代码中的任何一行比列宽（参见--columns），则表格将占据整个文本宽度，单元格内容将换行，相对单元格宽度由表格标题和表格主体之间分隔线的虚线数量决定。（例如，---|-将第一列设置为整个文本宽度的 3/4，第二列设置为整个文本宽度的 1/4。）另一方面，如果没有行比列宽更宽，则单元格内容将不会换行，单元格将根据其内容调整大小。

示例：

```
: Demonstration of pipe table syntax.

| Right | Left | Default | Center |
| ----: | :--- | ------- | :----: |
|    12 | 12   | 12      |   12   |
|   123 | 123  | 123     |  123   |
|     1 | 1    | 1       |   1    |

: 在 Markdown 代码中列垂直对齐的管道表格

| fruit  | price |
| ------ | ----: |
| apple  |  2.05 |
| pear   |  1.37 |
| orange |  3.09 |

: 在 Markdown 代码中没有列垂直对齐的管道表格

fruit| price
-----|-----:
apple|2.05
pear|1.37
orange|3.09

: 没有表头的管道表格

|
| ------ | ----: |
| apple  |  2.05 |
| pear   |  1.37 |
```

```
| orange |  3.09 |
```

表 2.14: Demonstration of pipe table syntax.

Right	Left	Default	Center
12	12	12	12
123	123	123	123
1	1	1	1

表 2.15: 在 Markdown 代码中列垂直对齐的管道表格

fruit	price
apple	2.05
pear	1.37
orange	3.09

表 2.16: 在 Markdown 代码中没有列垂直对齐的管道表格

fruit	price
apple	2.05
pear	1.37
orange	3.09

表 2.17: 没有表头的管道表格

apple	2.05
pear	1.37
orange	3.09

2.9.6 Caveats

1. 在 Pandoc 表格中插入图片是支持的，而且相对简单。你可以使用标准的 Markdown 图片语法在表格的单元格中插入图片。

2. 在 Pandoc 中，直接在 Markdown 表格的单元格内插入多行围栏式代码块是不支持的。Pandoc 的标准 Markdown 表格语法（如 Pipe Tables、Simple Tables 和 Grid Tables）对单元格内容有一些限制。它们通常期望单元格内容是单行的，或者至少是不包含像围栏式代码块那样需要多行且有特定格式的块级元素。如果你尝试这样做，Pandoc 通常会将其解析为表格中断，或者将代码块视为表格之外的独立内容，导致表格结构被破坏。在 Pandoc Markdown 表格中直接插入多行围栏式代码块是不推荐且通常不可行的。Pandoc 的表格语法设计目的不是为了容纳这种复杂的块级内容。

3. 在 Pandoc 的 Markdown 表格中处理多行中文内容时，容易出现换行错乱、对齐错误或单元格溢出等问题。以下是具体原因和解决方案：

问题原因：

- 换行符不兼容：Pandoc 的 Markdown 表格默认用 `<br>` 或空格换行，但中文段落换行可能被忽略。
- 中文字符宽度计算错误：中文字符宽度是西文的 2 倍，导致自动列宽计算失效。
- 表格语法限制：Pandoc 的 pipe_tables（| 分隔表格）对多行支持较弱，尤其是跨行内容。

解决方法：

- 确保分隔线 `|---|` 的宽度足够（如 `|--------|`）。
- 在网格表格中用空行分隔段落。

示例：

```
: 中文多行表格格式错误

+----------------+----------------+
| 列1            | 列2            |
+================+================+
| 第一行         | 中文内容       |
| 第二行         | 换行后的内容   |
+----------------+----------------+

: 在网格表格中用空行分隔段落

+----------------+----------------+
| 列1            | 列2            |
+================+================+
| 第一行         | 中文内容       |
|                |                |
| 第二行         | 换行后的内容   |
+----------------+----------------+
```

表 2.18: 中文多行表格格式错误

列 1	列 2
第一行第二行	中文内容换行后的内容

表 2.19: 在网格表格中用空行分隔段落

列 1	列 2
第一行 第二行	中文内容 换行后的内容

2.10 内联格式 Inline Formatting

2.10.1 强调

- 斜体：使用 * 或 _ 包裹
- 粗体：使用 ** 或 __ 包裹
- * 或 _ 被空格包围或用反斜杠转义不会触发强调

例如：

```
This text is _emphasized with underscores_, and this
is *emphasized with asterisks*.

This is **strong emphasis** and __with underscores__.

This is * not emphasized *, and \*neither is this\*.
```

This text is *emphasized with underscores*, and this is *emphasized with asterisks*.

This is **strong emphasis** and **with underscores**.

This is * not emphasized *, and *neither is this*.

2.10.2 扩展：intraword_underscores

由于 _ 有时会在单词和标识符内部使用，因此 pandoc 不会将 _ 字母数字字符包围的解释为强调标记。如果只想强调单词的一部分，请使用 *：

```
feas*ible*, not feas*able*.
```

---

feas*ible*, not feas*able*.

Pandoc 默认是启用该扩展的，如果要想将单词内部的 _ 解析强调语法，需要取消该扩展：`-f markdown-intraword_underscores`。

2.10.3 扩展：strikeout

要用水平线划掉一段文本，请以 ~~ 开始和结束该部分。

在转化为 LaTeX/PDF 时，unline 类会转化成 `\st{...}`（`soul` 宏包命令），对于中文是不起作用的。在 Stenciler 模板中，使用了 `CJKfntef` 宏包的 `CJKsout` 命令重新定义。

例如，

```
Lorem Ipsum is simply dummy text of the printing and typesetting
   industry. ~~Lorem Ipsum has been the industry's standard dummy
   text ever since the 1500s, when an unknown printer took a galley
   of type and scrambled it to make a type specimen book.~~ It has
   survived not only five centuries, but also the leap into
   electronic typesetting, remaining essentially unchanged. ~~It was
    popularised in the 1960s with the release of Letraset sheets
   containing Lorem Ipsum passages, and more recently with desktop
   publishing software like Aldus PageMaker including versions of
   Lorem Ipsum.~~
```

```
Lorem Ipsum，也称乱数假文或者哑元文本， 是印刷及排版领域所常用的虚拟文字。~~由于曾经一台匿名的打印机刻意打乱了一盒印刷字体从而造出一本字体样品书，Lorem Ipsum从西元15世纪起就被作为此领域的标准文本使用。~~ 它不仅延续了五个世纪，还通过了电子排版的挑战，其雏形却依然保存至今。~~在1960年代，“Leatraset”公司发布了印刷着Lorem Ipsum段落的纸张，从而广泛普及了它的使用。最近，计算机桌面出版软件“Aldus PageMaker”也通过同样的方式使Lorem Ipsum落入大众的视野。~~
```

Lorem Ipsum is simply dummy text of the printing and typesetting industry. ~~Lorem Ipsum has been the industry's standard dummy text ever since the 1500s, when an unknown printer took a galley of type and scrambled it to make a type specimen book.~~ It has survived not only five centuries, but also the leap into electronic typesetting, remaining essentially unchanged. ~~It was popularised in the 1960s with the release of Letraset sheets containing Lorem Ipsum passages, and more recently with desktop publishing software like Aldus PageMaker including versions of Lorem Ipsum.~~

Lorem Ipsum，也称乱数假文或者哑元文本，是印刷及排版领域所常用的虚拟文字。~~由于曾经一台匿名的打印机刻意打乱了一盒印刷字体从而造出一本字体样品书，~~Lorem ~~Ipsum 从西元 15 世纪起就被作为此领域的标准文本使用~~。 它不仅延续了五个世纪，还通过了电子排版的挑战，其雏形却依然保存至今。~~在 1960 年代，“Leatraset” 公司发布了印刷着 Lorem Ipsum 段落的纸张，从而广泛普及了它的使用。最近，计算机桌面出版软件 “Aldus PageMaker” 也通过同样的方式使 Lorem Ipsum 落入大众的视野~~。

2.10.4 扩展：superscript 和 subscript

上标可以用字符包围上标文本来书写^；下标也可以用字符包围下标文本来书写~。例如，

```
H~2~O is a liquid.  2^10^ is 1024.
```

H_2O is a liquid. 2^{10} is 1024.

---

^...^ 或之间的文本 ~...~ 不能包含空格或换行符。如果上标或下标文本包含空格，则必须使用反斜杠转义这些空格。（这是为了防止在日常使用 ~ 和时意外地将文本

变为上标或下标 ^，以及与脚注产生不良交互。）因此，如果您希望下标中包含字母 P 和 “a cat”，请使用 `P~a\ cat~`，而不是 `P~a cat~`。

2.10.5 抄录（代码）

- 要抄录一小段文本，请将其放在反引号内
- 如果抄录文本包含反引号，请使用双反引号
- 反斜杠转义符（和其他 Markdown 结构）在抄录环境中不起作用

```
What is the difference between `>>=` and `>>`?

Here is a literal backtick `` ` ``.

This is a backslash followed by an asterisk: `\*`.
```

What is the difference between >>= and >>?

Here is a literal backtick `.

This is a backslash followed by an asterisk: *.

2.10.6 扩展：`inline_code_attributes`

可以将属性附加到逐字文本，就像围栏式代码块一样：

```
<!-- pandoc --list-highlight-languages 查看支持的语言 -->
| 语言     | HelloWorld 语句                    |
| :--:    | :-----                            |
| C       | `printf("hello,world")`{.c}       |
| C++     | `cout << "hello,world"`{.cpp}     |
| Java    | `println("hello,world")`{.java}   |
| Python  | `print('hello,world')`{.python}   |
```

语言	HelloWorld 语句
C	`printf("hello,world")`

语言	HelloWorld 语句
C++	`cout << "hello,world"`
Java	`println("hello,world")`
Python	`print('hello,world')`

2.10.7 下划线

要为文本添加下划线，请使用以下 underline 类。

在转化为 LaTeX/PDF 时，unline 类会转化成 `\ul{...}`（soul 宏包命令），对于中文是不起作用的。在 Stenciler 模板中，使用了 CJKfntef 宏包的 CJKunderline 命令重新定义。

示例：

```
Lorem Ipsum is simply dummy text of the printing and typesetting
   industry. [Lorem Ipsum has been the industry's standard dummy
   text ever since the 1500s, when an unknown printer took a galley
   of type and scrambled it to make a type specimen book.]{.
   underline} It has survived not only five centuries, but also the
   leap into electronic typesetting, remaining essentially unchanged
   . [It was popularised in the 1960s with the release of Letraset
   sheets containing Lorem Ipsum passages, and more recently with
   desktop publishing software like Aldus PageMaker including
   versions of Lorem Ipsum.]{.underline}

Lorem Ipsum，也称乱数假文或者哑元文本，是印刷及排版领域所常用的虚拟
   文字。[由于曾经一台匿名的打印机刻意打乱了一盒印刷字体从而造出一本
   字体样品书，Lorem Ipsum从西元15世纪起就被作为此领域的标准文本使
   用。]{.underline} 它不仅延续了五个世纪，还通过了电子排版的挑战，
   其雏形却依然保存至今。[在1960年代，“Leatraset”公司发布了印刷着
   Lorem Ipsum段落的纸张，从而广泛普及了它的使用。最近，计算机桌面出
   版软件“Aldus PageMaker”也通过同样的方式使Lorem Ipsum落入大众的
   视野。]{.underline}
```

Lorem Ipsum is simply dummy text of the printing and typesetting industry. Lorem

Ipsum has been the industry's standard dummy text ever since the 1500s, when an unknown printer took a galley of type and scrambled it to make a type specimen book. It has survived not only five centuries, but also the leap into electronic typesetting, remaining essentially unchanged. It was popularised in the 1960s with the release of Letraset sheets containing Lorem Ipsum passages, and more recently with desktop publishing software like Aldus PageMaker including versions of Lorem Ipsum.

Lorem Ipsum，也称乱数假文或者哑元文本，是印刷及排版领域所常用的虚拟文字。由于曾经一台匿名的打印机刻意打乱了一盒印刷字体从而造出一本字体样品书，Lorem Ipsum 从西元 15 世纪起就被作为此领域的标准文本使用。 它不仅延续了五个世纪，还通过了电子排版的挑战，其雏形却依然保存至今。在 1960 年代，“Leatraset” 公司发布了印刷着 Lorem Ipsum 段落的纸张，从而广泛普及了它的使用。最近，计算机桌面出版软件 “Aldus PageMaker” 也通过同样的方式使 Lorem Ipsum 落入大众的视野。

2.10.8 Small Caps

要编写小型大写字母，请使用以下 smallcaps 类：

```
[Small caps]{.smallcaps}
```

Small caps

2.10.9 高亮显示

要突出显示文本，请使用以下 mark 类。

在转化为 LaTeX/PDF 时，unline 类会转化成 `\hl{...}`（`soul` 宏包命令），对于中文是不起作用的。`CJKfntef` 宏包不支持中文高亮显示。在 Stenciler 模板中，使用了 `CJKfntef` 宏包绘制字高宽度的下划线来替代。（见https://tex.stackexchange.com/questions/75019/is-there-any-solution-for-highlighting-text-in-cjk）

示例：

```
Lorem Ipsum is simply dummy text of the printing and typesetting
   industry. [Lorem Ipsum has been the industry's standard dummy
   text ever since the 1500s, when an unknown printer took a galley
   of type and scrambled it to make a type specimen book.]{.mark} It
    has survived not only five centuries, but also the leap into
   electronic typesetting, remaining essentially unchanged. [It was
   popularised in the 1960s with the release of Letraset sheets
   containing Lorem Ipsum passages, and more recently with desktop
   publishing software like Aldus PageMaker including versions of
   Lorem Ipsum.]{.mark}

Lorem Ipsum，也称乱数假文或者哑元文本， 是印刷及排版领域所常用的虚拟
   文字。[由于曾经一台匿名的打印机刻意打乱了一盒印刷字体从而造出一本
   字体样品书，Lorem Ipsum从西元15世纪起就被作为此领域的标准文本使
   用。]{.mark} 它不仅延续了五个世纪，还通过了电子排版的挑战，其雏形
   却依然保存至今。[在1960年代，“Leatraset”公司发布了印刷着Lorem
   Ipsum段落的纸张，从而广泛普及了它的使用。最近，计算机桌面出版软件
   “Aldus PageMaker”也通过同样的方式使Lorem Ipsum落入大众的视野。
   ]{.mark}
```

Lorem Ipsum is simply dummy text of the printing and typesetting industry. Lorem Ipsum has been the industry's standard dummy text ever since the 1500s, when an unknown printer took a galley of type and scrambled it to make a type specimen book. It has survived not only five centuries, but also the leap into electronic typesetting, remaining essentially unchanged. It was popularised in the 1960s with the release of Letraset sheets containing Lorem Ipsum passages, and more recently with desktop publishing software like Aldus PageMaker including versions of Lorem Ipsum.

Lorem Ipsum，也称乱数假文或者哑元文本，是印刷及排版领域所常用的虚拟文字。由于曾经一台匿名的打印机刻意打乱了一盒印刷字体从而造出一本字体样品书，Lorem Ipsum 从西元 15 世纪起就被作为此领域的标准文本使用。 它不仅延续了五个世纪，还通过了电子排版的挑战，其雏形却依然保存至今。在 1960 年代，“Leatraset” 公司发布了印刷着 Lorem Ipsum 段落的纸张，从而广泛普及了它的使用。最近，计算机桌面出版软件 “Aldus PageMaker” 也通过同样的方式使 Lorem Ipsum 落入大众的视野。

2.11 链接 Links

使用链接有下面几种形式：

- 自动链接：`<URL>`
- 内联链接：`[链接文本](URL)` 或 `[链接文本](URL "链接标题")`
- `[链接文本][链接标签]`
- 隐式链接 (“链接文本” 和” 链接标题” 相同)：`[链接文本][]`
- 快捷链接 (“链接文本” 和” 链接标题” 相同)：`[链接文本]`
- `[链接文本](#标题标识符)` 或 `[标题]`[5]

“链接标签” 的定义：

- `[链接标签]: URL`
- `[链接标签]: URL "链接标题"`
- `[链接标签]: URL (链接标题)`
- `[链接标签]: <URL>`

2.11.1 自动链接 Automatic Links

如果将 URL 或电子邮件地址放在尖括号中，它将变成链接：

```
<https://daringfireball.net/projects/markdown/syntax>

<https://www.markdownguide.org/>
```

---

https://daringfireball.net/projects/markdown/syntax

https://www.markdownguide.org/

2.11.2 内联链接 Inline Links

内联链接由方括号内的链接文本和括号内的 URL[6] 组成。(URL 后面也可以跟着链接标题，并用引号引起来。)

[5]标题标识符可以是 Pandoc 按一定规则自动生成，也可以手动定义。

[6]URL 可以是网址也可以是本地路径。

```
The [Markdown Guide](https://www.markdownguide.org/ "Click to Visit
    Markdown Guide") is a free and open-source reference guide that
    explains how to use Markdown, the simple and easy-to-use markup
    language you can use to format virtually any document.
```

The Markdown Guide is a free and open-source reference guide that explains how to use Markdown, the simple and easy-to-use markup language you can use to format virtually any document.

方括号和圆括号之间不能有空格。链接文本可以包含格式（例如强调），但标题不能。

内联链接中的电子邮件地址无法自动检测，因此必须加上前缀 `mailto`：

```
[Write me!](mailto:sam@green.eggs.ham)
```

2.11.3 引用链接 Reference Links

显式引用链接包含两部分：链接本身和链接定义，后者可能出现在文档的其他位置（链接之前或之后）。

链接由方括号内的链接文本和方括号内的标签组成。（除非启用了扩展 `spaced_reference_links`，否则两者之间不能有空格。）

链接定义由方括号内的标签、冒号和空格、URL 以及可选的（空格后）链接标题（用引号或括号括起来）组成。

标签不能被解析为文献引用（假设启用了扩展 `citations`）：文献引用优先于链接标签。

URL 可以选择性地用尖括号括起来。

标题可以放在下一行。

请注意，链接标签不区分大小写。

在隐式引用链接中，第二对括号为空。

以下是一些示例：

```
- 设计哲学: [Markdown Philosophy]
- 基本语法: [Markdown Syntax]
- 基本语法: [Basic Syntax][BASIC]<!-- 请注意, 链接标签不区分大小写
   -->
- 元素: [Block Elements]
- Markdown 指南: [Markdown Guide]
- 扩展语法: [Extended Syntax]
- 速查表: [Cheat Sheet][]<!-- 在隐式引用链接中, 第二对括号为空 -->

[Markdown Philosophy]: https://daringfireball.net/projects/markdown/
   syntax#philosophy  "Philosophy of Markdown"
[Block Elements]: https://daringfireball.net/projects/markdown/
   syntax#block 'Block Elements'
[Markdown Syntax]: https://daringfireball.net/projects/markdown/
   syntax
[Markdown Guide]: https://www.markdownguide.org/ (Markdown Guide)
[Extended Syntax]: <https://www.markdownguide.org/extended-syntax/>
[basic]: https://www.markdownguide.org/basic-syntax/
[Cheat Sheet]: https://www.markdownguide.org/cheat-sheet/
```

- 设计哲学: Markdown Philosophy
- 基本语法: Markdown Syntax
- 基本语法: Basic Syntax
- 元素: Block Elements
- Markdown 指南: Markdown Guide
- 扩展语法: Extended Syntax
- 速查表: Cheat Sheet

在 Markdown.pl Markdown 和大多数其他实现中，引用链接定义不能出现在嵌套结构中，例如列表项或块引用。Pandoc 取消了这一限制。因此，以下代码在 Pandoc 中可以正常工作，但在大多数其他实现中则不行：

```
> The most visited search egnine is [google].

> The top 3 search engines:

  - [google]
  - yandex
  - baidu

[google]: https://www.google.com
```

The most visited search egnine is google.

The top 3 search engines:

- google
- yandex
- baidu

2.11.4 扩展：shortcut_reference_links

在快捷引用链接中，第二对括号可以完全省略：

```
See [my website].

[my website]: http://foo.bar.baz
```

2.11.5 内部链接 Internal Links

要链接到同一文档的其他部分，请使用自动生成的标识符（请参阅标题标识符）。例如：

```
See the [Introduction].
```

或者

```
See the [Introduction](#introduction).
```

或者

```
See the [Introduction].

[Introduction]: #introduction
```

内部链接目前支持 HTML 格式（包括 HTML 幻灯片和 EPUB）、LaTeX 和 ConTeXt。

2.12 图片 Images

在链接语法前加上 ！就是图片插入语法。“链接文本” 将用作图片的替代文本，图片的标题是 “链接文本” 内容，“链接标题” 在鼠标悬停在图片上时显示的文本（见扩展：*implicit_figures*）。

如果指定了图片的标题，则图片自动居中显示；否则，左侧可能缩进。

在使用 Pandoc 将 Markdown 文档转换为 PDF（通过 LaTeX）时，图片插入位置默认是浮动的（floating），并且将自动将图片宽度限制在文本宽度（`textwidth`）内，如：

```
<!-- 使用图片：在链接前加上感叹号！ -->

<!-- 内联形式："标题" 在鼠标悬停在图上时显示 -->
![内联形式图片](examples/pandoc-flavored-markdown/../../images/tux.
   png "Tux")

<!-- 引用形式 -->
![引用形式图片][img-ref]

[img-ref]: examples/pandoc-flavored-markdown/../../images/tux.png "
   Tux"

<!-- 隐式形式 -->
![隐式形式图片][]

[隐式形式图片]: examples/pandoc-flavored-markdown/../../images/tux.
   png "Tux"

<!-- 快捷形式 -->
![快捷形式图片]

[快捷形式图片]: examples/pandoc-flavored-markdown/../../images/tux.
   png "Tux"
```

图 2.1: 内联形式图片

图 2.2: 引用形式图片

图 2.3: 隐式形式图片

图 2.4: 快捷形式图片

2.12.1 扩展: implicit_figures

如果图片带有非空的 alt 文本，且单独出现在段落中，则该图片将被渲染为带有标题的图形。图片的 alt 文本将用作标题。

```
![This is the caption](/url/of/image.png)
```

如何渲染取决于输出格式。某些输出格式（例如 RTF）尚不支持图形。在这些格式中，您只会在段落中看到一张图片，没有标题。

如果你只是想要一个普通的内联图片，请确保它不是段落中唯一的内容。一种方法是在图片后插入一个不间断的空格:

```
![This image won't be a figure](/url/of/image.png)\
```

请注意，在 reveal.js 幻灯片中，段落中具有该类的图像本身 r-stretch 将填满屏幕，并且标题和图形标签将被省略。

2.12.2 扩展: link_attributes

图片的 width 和 height 属性会被特殊处理。如果不带单位使用，则单位被假定为像素（pixels）。但是，可以使用以下任何单位标识符: px、cm、mm、in、inch 和 %。**数字和单位之间不得有任何空格**。例如:

```
![](chapters/pandoc-flavored-markdown/file.jpg){ width=50% }
```

- 尺寸可能会转换为与输出格式兼容的形式（例如，将 HTML 转换为 LaTeX 时，以像素为单位的尺寸将转换为英寸）。像素与物理测量值之间的转换受 --dpi 选项影响（默认情况下，假定为 96 dpi，除非图像本身包含 dpi 信息）。
- 单位 % 通常与某个可用空间相关。例如，上面的示例将渲染成以下内容。
  - HTML:
  - LaTeX: \includegraphics[width=0.5\textwidth,height=\textheight]{file.jpg}（如果您使用自定义模板，则需要 graphicx 按照默认模板进行配置。）
  - ConTeXt:\externalfigure[file.jpg][width=0.5\textwidth]
- 一些输出格式具有类（ConTeXt）或唯一标识符（LaTeX \caption）或两者（HTML）。

- 当未指定 width 或 height 属性时，后备方法是查看图像分辨率和图像文件中嵌入的 dpi 元数据。

示例：

```
![图片属性：width=50%](examples/pandoc-flavored-markdown/../../
   images/tux.png){ width=50% }

![图片属性：width=150px](examples/pandoc-flavored-markdown/../../
   images/tux.png){ width=150px }

![图片属性：height=160px](examples/pandoc-flavored-markdown/../../
   images/tux.png){ height=160px }
```

图 2.5: 图片属性：width=50%

图 2.6: 图片属性：width=150px

图 2.7: 图片属性：height=160px

2.13 Divs 和 Spans

通过使用 native_divs[7] 和 native_spans[8] 扩展，可以在 Markdown 中使用 HTML 语法来创建 Pandoc AST 中的原生 Div 和 Span 元素。不过，还有更简洁的语法可用：

2.13.1 扩展：fenced_divs

允许使用特殊的围栏语法来创建原生 Div 块。Div 块以至少包含三个连续冒号（:::）的围栏开始，后面跟着一些属性。属性后可以选择性地再跟一串连续冒号。

[7] https://pandoc.org/MANUAL.html#extension-native_divs

[8] https://pandoc.org/MANUAL.html#extension-native_spans

`commonmark` 解析器不允许在属性后使用冒号。

属性语法与围栏式代码块中的完全相同（参见扩展：*fenced_code_attributes*）。与围栏式代码块类似，可以使用花括号中的属性或单个不带花括号的单词（将被视为类名）。Div 块以另一行包含至少三个连续冒号的围栏结束。围栏 Div 应通过空行与前后块分隔。

在转换成 LaTeX 时，Div 不起任何作用；如果需要特殊效果，需要自定义过滤器来处理，见过滤器实例：启用 *LaTeX* 环境。

示例：

```
::::: {#special .sidebar}
这是一个段落。

还有一个。
:::::
```

围栏 Div 可以嵌套。开头的围栏因必须包含属性而有所区分：

```
::: Warning ::::::
这是一个警告。

::: Danger
这是警告中的一个警告。
:::
::::::::::::::::::
```

不带属性的围栏始终是结束围栏。与围栏式代码块不同，结束围栏中的冒号数量不必与开头围栏中的数量匹配。不过，为了视觉清晰，使用不同长度的围栏来区分嵌套 Div 与其父级 Div 是很有帮助的。

2.13.2 扩展：bracketed_spans

如果一个括号中的内联序列（如同开始链接时使用的语法）紧接着跟有属性，它将被视为带有属性的 Span：

```
[这是 *一些文本*]{.class key="val"}
```

通过 Divs 和 Spans 的 id 属性，可以文档内部实现链接（见链接 *Links*）：

```
[链接文本](#标识符)
```

2.14 脚注 Footnotes

2.14.1 扩展：footnotes

Pandoc 的 Markdown 允许使用脚注，使用以下语法：

```
Here is a footnote reference,[^1] and another.[^longnote]

[^1]: Here is the footnote.

[^longnote]: Here's one with multiple blocks.

    Subsequent paragraphs are indented to show that they
belong to the previous footnote.

        { some.code }

    The whole paragraph can be indented, or just the first
    line.  In this way, multi-paragraph footnotes work like
    multi-paragraph list items.

This paragraph won't be part of the note, because it
isn't indented.
```

脚注引用中的标识符不得包含空格、制表符、换行符或字符 ^、[、或]。这些标识符仅用于将脚注引用与注释本身关联起来；在输出中，脚注将按顺序编号。

脚注本身不必放在文档末尾。它们可以出现在除其他块元素（列表、块引用、表格等）之外的任何位置。每个脚注都应与周围内容（包括其他脚注）用空行分隔。

2.14.2 扩展：inline_notes

行内脚注也是允许的（不过，与常规注释不同，行内脚注不能包含多个段落）。语法如下：

```
Here is an inline note.^[Inline notes are easier to write, since
you don't have to pick an identifier and move down to type the
```

```
note.]
```

行内脚注和常规脚注可以自由混合。

2.14.3 多次引用同一脚注

在多次引用同一脚注时，会出现该脚注被重复输出[9]。解决方法：直接使用 `\footnotemark` 和 `\footnotetext` 命令来实现；也可以通过设置脚注标签和特殊链接格式（需要过滤器处理）来解决。

Markdown 示例如下：

```
This is the first note[^note1], this is the second note[^note2],
   this is the reference to the first note^[](#notelabel1)^.

[^note1]: [foo]{#notelabel1}
[^note2]: bar
```

使用 Panflute 编写过滤器（见过滤器 *Filters*）：

```
#!/usr/bin/env python3

import Panflute as pf

def action(elem, doc):
    # Process Link elements
    if isinstance(elem, pf.Link) and elem.url.startswith('#'):
        labelid = elem.url[1:]  # Remove '#' to get labelid
        if not elem.content:  # Empty link, e.g., [](#labelid)
            if doc.format == 'latex':
                # Convert to LaTeX \ref{labelid}
                return pf.RawInline(f'\\ref{{{labelid}}}', format='
                   latex')
            elif doc.format == 'html':
                # Convert to HTML <a href="#labelid">labelid</a>
                return pf.RawInline(f'<a href="#{labelid}">{labelid
                   }</a>', format='html')
    return elem


if __name__ == '__main__':
    pf.run_filter(action, doc=None)
```

[9]见 https://github.com/jgm/pandoc/issues/1603 。

2.15 HTML 代码

2.15.1 扩展：raw_html

可以直接在文档中插入 HTML 代码（除了代码块等，其中 <, > 和 & 符不会被翻译）。因为标准 Markdown 允许插入 HTML，但是 Pandoc 为了可以根据需要禁用 HTML，将其设计成一个扩展。

2.15.2 扩展：markdown_in_html_blocks

使用 markdown_strict 格式的时候，HTML 代码中的 Markdown 语法不会被翻译，但是使用 Pandoc 的 Markdown 格式 markdown 格式时，HTML 代码中的 Markdown 语法也会被翻译，但是有一个例外，HTML 代码 <script> 和 <style> 标签中的 Markdown 语法也不会被翻译。

例如，Pandoc 会将

```
<table>
<tr>
<td>*one*</td>
<td>[a link](https://google.com)</td>
</tr>
</table>
```

转换成：

```
<table>
<tr>
<td><em>one</em></td>
<td><a href="https://google.com">a link</a></td>
</tr>
</table>
```

2.16 LaTeX/TeX 代码

2.16.1 扩展：raw_tex

除了原始 HTML 之外，Pandoc 还允许将原始 LaTeX、TeX 和 ConTeXt 代码包含在文档中。内联 TeX 命令将被保留，并原封不动地传递给 LaTeX 和 ConTeXt 编写器。

因此，例如，您可以使用 LaTeX 来包含 BibTeX 引用：

```
This result was proved in \cite{jones.1967}.
```

请注意，在 LaTeX 环境中，例如

```
\begin{tabular}{|l|l|}\hline
Age & Frequency \\ \hline
18--25  & 15 \\
26--35  & 33 \\
36--45  & 22 \\ \hline
\end{tabular}
```

开始和结束标签之间的材料将被解释为原始 LaTeX，而不是 Markdown。

第三章　数学公式 Mathematical Expressions

数学公式包含两种：

Inline math 即**行内公式**或**内联公式**，指的是将数学公式嵌入到文本段落的行内，与其他文本内容混合排列。在 LaTeX 中，行内公式通常用单个美元符号 `$` 包裹。

Displayed math 即**行间公式**或**显示公式**，指的是将数学公式单独占据一行显示，通常会居中排列，并且比行内公式具有更大的视觉突出性。在 LaTeX 中，行间公式通常用双美元符号 `$$` 包裹。

总而言之，行内公式用于在文本中简短地插入数学表达式，而行间公式则用于更重要或更复杂的数学公式，使其在文档中更加醒目。

Pandoc 生成 HTML 或 EPUB 需要使用 MathML 或 MathJax，如：

```
pandoc -s --mathjax -o document.epub document.md
```

Pandoc 生成 docx 时将使用 OMML 数学标记来呈现：

```
pandoc -s -o document.docx document.md
```

Pandoc 生成 LaTeX 时将逐字显示，周围环绕着`\(...\)`（行内公式）或`\[...\]`（行间公式）。

Pandoc 生成 PDF 时先调用 LaTeX 引擎。

3.1　行内公式 Inline math

```
Euler's formula is remarkable: $e^{i\pi} + 1 = 0$.
```

Euler's formula is remarkable: $e^{i\pi} + 1 = 0$.

3.2 行间公式 Display Math

```
When $a \ne 0$, there are two solutions to $ax^2 + bx + c = 0$ and
   they are

$$ x = \dfrac{-b \pm \sqrt{b^2-4ac}}{2a} $$
```

When $a \ne 0$, there are two solutions to $ax^2 + bx + c = 0$ and they are

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

第四章　交叉引用 Cross Referencing

4.1　使用 pandoc-crossref

pandoc-crossref 主要用于交叉引用图表、公式、章节、表格等，但不支持直接交叉引用参考文献（如 `[@citekey]` 生成的文献引用）。

4.1.1　安装 pandoc 和 pandoc-crossref

要想使用 pandoc-crossref，需要安装相互兼容的 pandoc 和 pandoc-crossref；否则 Pandoc 会提示兼容性问题。如：pandoc-3.6.4 和 pandoc-crossref-v0.3.18.2。

- pandoc 下载地址：https://github.com/jgm/pandoc/releases
- pandoc-crossref 下载地址：https://github.com/lierdakil/pandoc-crossref/releases

解压缩 pandoc 可执行文件到一个目录中，如 `~/.local/bin/pandoc`。解压缩 pandoc-crossref 可执行文件到 `~/.local/bin/pandoc-crossref` 中. 帮助文档拷贝到 `~/.local/share/man/man1`中，目录结构如下：

```
$ tree ~/.local/share/man
/home/USERNAME/.local/share/man
└── man1
    ├── pandoc.1.gz
    ├── pandoc-crossref.1
    ├── pandoc-lua.1.gz
    └── pandoc-server.1.gz
```

4.1.2　运行 pandoc-crossref

首先，需要设置环境变量 PATH：

```
export PATH=~/.local/bin:$PATH
```

在 macOS 中，首次运行 `pandoc --filter pandoc-crossref` 时会提示：“无法打开 “pandoc-crossref”，因为无法验证开发者。” 点击 “取消”，然后在 “系统偏好设置-安全性与隐私-通用” 中 “仍然允许” 通过验证。

4.1.3 直接转换成 PDF 时的错误处理

直接转换时出现以下错误：

```
Error producing PDF.
! Undefined control sequence.
l.340 (\passthrough
```

可以先转换成.tex 文件，然后在文件的导言区中加入：

```
\usepackage{listings}
\newcommand{\passthrough}[1]{#1}
```

或者：添加 `-M listings` 选项[1]。

4.2 引用章节 Referencing Sections

在文本中定义章节标签 `{#sec:label}`，然后引用该标签 `[@sec:label]`：

```
# This is a chapter {#sec:ch}

## This is the first section {#sec:a}

Lorem ipsum dolor sit amet, consectetur adipiscing elit.

## This is the second section {#sec:b}

Sed iaculis pellentesque tempor. Etiam fringilla orci ut est
   efficitur, vitae iaculis odio convallis.

See [@sec:a] and [@sec:b].
```

编译时必须使用 `--number-sections`，如：

```
pandoc --number-sections --filter pandoc-crossref -o output.pdf
   input.md
```

[1] pandoc-crossref と Eisvogel を併用する

使用 pandoc-crossref 的 [@sec:label] 只能显示章节编号；而 Pandoc 的扩展：*implicit_header_references* 能够显示章节名称。

4.3 引用图片 Referencing Figures

在文本中定义图片标签 {#fig:label}，然后引用该标签 [@fig:label]：

```
![alt text](/path/to/image){#fig:label}
```

如果需要加入属性：

```
![alt text](/path/to/image){#fig:label width=32px}
```

4.4 引用表格 Referencing Tables

在文本中定义表格标签 {#tbl:label}，然后引用该标签 [@tbl:label]：

```
 a   b   c
--- --- ---
 1   2   3
 4   5   6

: Simple Table {#tbl:label}
```

Pandoc 表格并不支持属性设置（见https://stackoverflow.com/questions/41877612/pandoc-add-class-to-table-in-markdown），pandoc-crossref 采用了特殊处理方式（见https://github.com/jgm/pandoc/issues/6317#issuecomment-618978090）。

4.5 引用公式 Referencing Equations

在文本中定义公式标签 {#eq:label}，然后引用该标签 [@eq:label]：

```
$$ f(x) = x^2 + x + 1 $$ {#eq:label}
```

4.6 修改引用的前缀

单个引用：[@sec:label] 默认前缀是 sec.，显示如：sec. 1.5

合并引用：[@sec:label1; @sec:label2]，显示如：secs. 1.5, 1.6

去掉前缀：[-@sec:label]，显示如：1.5

自定义前缀：

```
---
secPrefix: ["Sect.", "Sects."]
---
```

类型	metadata
章节	secPrefix
图片	figPrefix
表格	tblPrefix
公式	eqnPrefix

第五章　文献引用 Citations

Pandoc 支持文献引用，常用格式为 BibTeX 或 BibLaTeX，结合 Markdown 或 LaTeX 进行引用。Pandoc 支持通过 CSL（Citation Style Language）文件自定义引用文献风格，用于格式化参考文献和引文。Pandoc 支持多种文献数据格式，常用的是 BibTeX (.bib) 或 BibLaTeX，也支持 JSON、YAML 等格式。

5.1　创建 BibTeX 文件

创建一个名为 `references.bib` 的文件，内容如下：

```
@book{knuth1997,
  author    = {Donald E. Knuth},
  title     = {The Art of Computer Programming, Volume 1:
     Fundamental Algorithms},
  year      = {1997},
  publisher = {Addison-Wesley},
  address   = {Reading, MA}
}

@article{lamport1994,
  author  = {Leslie Lamport},
  title   = {LaTeX: A Document Preparation System},
  journal = {Software: Practice and Experience},
  volume  = {24},
  number  = {3},
  year    = {1994},
  pages   = {307--317}
}
```

5.2 CSL 文件

Pandoc 使用 CSL 文件定义引文和参考文献的格式。你可以从 CSL GitHub 仓库[1]下载现成的 CSL 文件（如 APA、MLA、Chicago 等），或自定义 CSL 文件。例如：

- APA 风格：`apa.csl`
- Chicago 风格：`chicago-author-date.csl`

如果不指定 CSL 文件，Pandoc 默认使用 Chicago 作者-日期格式。

预览参考文献的风格：https://www.zotero.org/styles

以下是一些常见的 CSL 风格及其用途：

- **APA** (`apa.csl`)：心理学、社会科学常用，强调作者和年份。
- **MLA** (`modern-language-association.csl`)：人文学科，注重作者和标题。
- **Chicago** (`chicago-author-date.csl` 或 `chicago-note-bibliography.csl`)：历史学、文学等，分为作者-日期和笔记-书目两种。
- **IEEE** (`ieee.csl`)：工程技术领域，编号引文格式。

CSL 文件存储位置：

- Pandoc 有一个默认的数据目录，用于存储 CSL 文件、模板等资源。可以通过以下命令查看 Pandoc 的数据目录：

  ```
  pandoc --version
  ```

  输出中会显示 `Default user data directory`，通常是：

  - Linux/macOS: `~/.pandoc/` 或 `~/.local/share/pandoc/`
  - Windows: `C:\Users\<用户名>\AppData\Roaming\pandoc\`

- 你可以将 CSL 文件放在这个目录下的 `csl/` 子文件夹中，例如 `~/.pandoc/csl/style.csl`。

- 放在这里后，Pandoc 可以直接识别文件名，无需指定完整路径。例如：

  ```
  pandoc input.md --csl=style.csl -o output.pdf
  ```

[1] https://github.com/citation-style-language/styles

5.3 使用参考文献

5.3.1 指定.bib 文件和.csl 文件

可以通过命令行选项设置，如：

`--bibliography=references.bib --csl=apa.csl`

或也可以通过 YAML 文件指定，见引文渲染：

```
---
bibliography: references.bib
csl: apa.csl
---
```

5.3.2 文献引用语法

语法：

`@knuth1997`

`@lamport1994, p. 310`

渲染效果与 CSL 格式有关。

5.3.3 编译

运行以下 Pandoc 命令，将 Markdown 文件转换为目标格式（如 PDF 或 HTML）：

```
pandoc example.md --citeproc -o example.pdf
```

当与 `pandoc-crossref` 一起使用时，处理顺序很重要。通常情况下，你需要先运行 `pandoc-crossref`，然后再运行 `--citeproc`。这是因为 `pandoc-crossref` 使用了与 `--citeproc` 相似的 `[@...]` 引用语法，先运行 `pandoc-crossref` 可以避免两者之间的冲突。

```
pandoc example.md --filter pandoc-crossref --citeproc -o example.pdf
```

`pandoc-crossref` 并不能为参考文献建立链接。

5.3.4 `link-citations` 元数据

如果为 true，引文将超链接到对应的参考文献条目（仅适用于作者-日期和数字样式）。默认为 false。

在编译时加入：`-M link-citations=true` 或将 `link-citations=true` 写入元数据文件中（注意不是 defaults file）。

5.4 参考文献的位置

默认情况下，Pandoc 会将参考文献列表（由 `references.bib` 文件生成）自动放置在生成文档的末尾。这是学术写作的标准格式（如 APA、MLA、Chicago 等），因此也是 Pandoc 的默认行为。

如果你需要将参考文献放在其他位置怎么办？虽然不能通过一个简单的标记来改变位置，但有几种有效的方法可以实现你的需求：

5.4.1 方法一：使用 `div` 包裹引用（推荐且标准）

这是最 “Pandoc” 的方式。你可以在你的 Markdown 主文件（例如 `paper.md`）中，**精确地指定**你希望参考文献列表出现的位置。

1. 在你希望参考文献出现的地方，插入一个特定的 `div` 容器，并为其设置一个唯一的 ID，例如 `#refs`。

```
# 正文内容

这里是论文的正文内容，包含了各种引用 [@Author2020;
  @Scholar2018]。

## 参考文献

::: {#refs}
:::

## 附录

这里是附录内容。
```

2. 在使用 Pandoc 编译时，使用 `--citeproc` 或 `--filter=citeproc` 参数（它负责处理参考文献），Pandoc 会自动识别 `{#refs}` 这个特定的 div，并将生成

的参考文献列表填充到里面。

```
pandoc paper.md --bibliography=references.bib --citeproc -o
    paper.pdf
# 或者使用 CSL 样式文件
pandoc paper.md --bibliography=references.bib --csl chicago-
    author-date.csl --citeproc -o paper.pdf
```

关键点：{#refs} 这个 ID 是 citeproc 过滤器识别的一个特殊信号。当你使用它时，Pandoc 就不会再把参考文献放在文末，而是放在你指定的这个位置。

5.4.2 方法二：后处理文档结构（更灵活但复杂）

如果方法一不满足需求，或者你需要更复杂的控制，可以采用 “分体-合并” 的策略。

1. 将文档分成两部分：主文 (main.md) 和附录 (appendix.md)。
2. 单独为主文生成带参考文献的文档：

```
pandoc main.md --bibliography=references.bib --citeproc -o
    main_with_refs.md
```

这会生成一个包含了正确格式参考文献的 Markdown 文件。

3. 合并文件：

```
pandoc main_with_refs.md appendix.md -o final_paper.pdf
```

这种方法给你最大的灵活性，你可以把参考文献放在任何两个文件之间，但需要更多的步骤和文件管理。

5.4.3 方法三：使用 LaTeX 指令（仅适用于 PDF/LaTeX 输出）

如果你输出的目标是 PDF（通过 LaTeX 引擎），你可以在 Markdown 文件中直接使用 LaTeX 命令来控制位置。

在你的 Markdown 文件中插入以下代码：

```
# 正文内容

这里是论文的正文内容，包含了各种引用 [@Author2020; @Scholar2018]。

\begingroup
\renewcommand{\section}[2]{} % 可选：抑制参考文献部分的标题
```

```
\begin{thebibliography}{}
\end{thebibliography}
\endgroup

# 附录

这里是附录内容。
```

然后使用 Pandoc 编译：

```
pandoc paper.md --bibliography=references.bib --citeproc -o paper.
    pdf
```

Pandoc 的 `citeproc` 会将生成的参考文献列表填充到 `thebibliography` 环境中。注意：这种方法依赖于 LaTeX，因此只适用于输出格式为 PDF 的情况。

第六章　默认配置文件 Defaults files

Pandoc 的默认配置文件（defaults files）[1]是一个 YAML 格式[2]的配置文件，用于存储 Pandoc 转换文档时常用的选项和设置。它允许用户将命令行参数持久化保存，避免重复输入相同的参数，特别适合复杂的文档转换任务或团队协作场景。

`--defaults` 选项可用于指定一组选项，格式为 YAML 文件。默认文件可以放置在用户数据目录的 defaults 子目录中，并从任何目录使用。例如，可以创建一个指定写信默认设置的文件，将其保存为用户数据目录的 defaults 子目录中的 letter.yaml，然后从任何目录通过 `pandoc --defaults letter` 或 `pandoc -dletter` 调用这些默认设置。

当使用多个默认文件时，它们的设置将被合并。需要注意的是，对于可以重复的命令行参数（例如 `--metadata-file`、`--css`、`--include-in-header`、`--include-before-body`、`--include-after-body`、`--variable`、`--metadata`、`--syntax-definition`），命令行中指定的值将与默认文件中指定的值合并，而不是替换。

6.1　General Options

表 6.1: General Options

Command line	Default file
`--from markdown+emoji`	`from: markdown+emoji`
`--to markdown+hard_line_breaks`	`to: markdown+hard_line_breaks`
`--output foo.pdf`	`output-file: foo.pdf`
`--data-dir dir`	`data-dir: dir`

[1] https://pandoc.org/MANUAL.html#defaults-files

[2] YAML Ain't Markup Language，与 XML（标记语言）不同，YAML 避免使用标签（< >），而是依赖缩进和符号表示结构。

Command line	Default file
`--defaults file`	`defaults:` `  - file`

表格中只列举部分选项，详见 Pandoc 文档，下同。

默认文件中指定的选项始终优先于通过 `defaults:` 条目包含的其他文件中的选项。

6.2 阅读器选项 Reader Options

表 6.2: Reader Options

Command line	Default file
`--shift-heading-level-by -1`	`shift-heading-level-by: -1`
`--default-image-extension ".jpg"`	`default-image-extension: '.jpg'`
`--file-scope`	`file-scope: true`

`--citeproc \`
`--lua-filter count-words.lua \`
`--filter special.lua`
	`filters:`
	`  - citeproc`
	`  - count-words.lua`
	`  - type: json`
	`  path: special.lua`

`--metadata key=value \`
`--metadata key2`
	`metadata:`
	`  key: value`
	`  key2: true`

`--metadata-file meta.yaml`	`metadata-files:`
	`  - meta.yaml`
	OR
	`metadata-file: meta.yaml`

默认文件中指定的元数据值被解析为字面字符串文本（literal string text），而不是 Markdown。

如果过滤器具有 `.lua` 扩展名，则假定为 Lua 过滤器，否则为 JSON 过滤器。但也可以明确指定过滤器类型，如上所示。过滤器按指定顺序运行。要包含内置的 citeproc 过滤器，可以使用 `citeproc` 或 `{type: citeproc}`。

6.3 通用写入器选项

表 6.3: 通用写入器选项

Command line	Default file
`--standalone`	`standalone: true`
`--template letter`	`template: letter`

`--variable key=val \`
`--variable key2`
	`variables:`
	`  key: val`
	`  key2: true`

`--table-of-contents`	`table-of-contents: true`
`--toc`	`toc: true`
`--toc-depth 3`	`toc-depth: 3`
`--strip-comments`	`strip-comments: true`
`--no-highlight`	`highlight-style: null`
`--highlight-style kate`	`highlight-style: kate`
`--include-in-header inc.tex`	`include-in-header:` `  - inc.tex`
`--include-before-body inc.tex`	`include-before-body:` `  - inc.tex`
`--include-after-body inc.tex`	`include-after-body:` `  - inc.tex`

6.4 特定写入器选项

表 6.4: 特定写入器选项

Command line	Default file
`--self-contained`	`self-contained: true`
`--link-images`	`link-images: true`
`--html-q-tags`	`html-q-tags: true`

Command line	Default file
`--ascii`	`ascii: true`
`--reference-links`	`reference-links: true`
`--reference-location block`	`reference-location: block`
`--figure-caption-position=above`	`figure-caption-position: above`
`--table-caption-position=below`	`table-caption-position: below`
`--markdown-headings atx`	`markdown-headings: atx`
`--list-tables`	`list-tables: true`
`--top-level-division chapter`	`top-level-division: chapter`
`--number-sections`	`number-sections: true`
`--number-offset=1,4`	`number-offset: [1,4]`
`--listings`	`listings: true`
`--list-of-figures`	`list-of-figures: true`
`--lof`	`lof: true`
`--list-of-tables`	`list-of-tables: true`
`--lot`	`lot: true`
`--incremental`	`incremental: true`
`--slide-level 2`	`slide-level: 2`
`--section-divs`	`section-divs: true`
`--email-obfuscation references`	`email-obfuscation: references`
`--id-prefix ch1`	`identifier-prefix: ch1`
`--title-prefix MySite`	`title-prefix: MySite`
`--reference-doc my.docx`	`reference-doc: my.docx`
`--epub-cover-image cover.jpg`	`epub-cover-image: cover.jpg`
`--epub-title-page=false`	`epub-title-page: false`
`--epub-metadata meta.xml`	`epub-metadata: meta.xml`
`--split-level 2`	`split-level: 2`
`--chunk-template="%i.html"`	`chunk-template: "%i.html"`
`--pdf-engine xelatex`	`pdf-engine: xelatex`

`--pdf-engine-opt=--shell-escape`	`pdf-engine-opts:`
	`  - 'shell-escape`
	OR
	`pdf-engine-opt: '-shell-escape'`

6.5 引文渲染

表 6.5: 引文渲染

Command line	Default file
`--citeproc`	`citeproc: true`
`--bibliography logic.bib`	`bibliography: logic.bib`
`--csl ieee.csl`	`csl: ieee.csl`
`--citation-abbreviations ab.json`	`citation-abbreviations: ab.json`
`--natbib`	`cite-method: natbib`
`--biblatex`	`cite-method: biblatex`

`cite-method` 可以是 `citeproc`、`natbib` 或 `biblatex`。这仅影响 LaTeX 输出。如果要使用 citeproc 格式化引文，还应设置 `citeproc: true`。

如果需要控制 citeproc 处理与其他过滤器的相对顺序，应在过滤器列表中使用 citeproc（参见阅读器选项 *Reader Options*）。

6.6 HTML 中的数学渲染

表 6.6: HTML 中的数学渲染

Command line	Default file
`--mathjax`	`html-math-method:` `  method: mathjax`
`--mathml`	`html-math-method:` `  method: mathml`
`--webtex`	`html-math-method:` `  method: webtex`
`--katex`	`html-math-method:` `  method: katex`
`--gladtex`	`html-math-method:` `  method: gladtex`

除上述值外，`method` 还可以取值 `plain`。

如果命令行选项接受 URL 参数，可以在 `html-math-method:` 中添加 `url:` 字段。

第七章　使用元数据 Metadata

在 Pandoc 中，YAML 元数据是指文档开头的一段 YAML 格式的内容，用于描述文档的属性和元信息。通过读取 Pandoc 标记文件中的 YAML 元数据，我们可以获取文档的各种属性，如标题、作者、日期等，以及自定义的元信息。

当 Pandoc 生成独立文档时，会利用标题、作者等元数据来填充部分信息。这些数据通常不会在 Markdown 源文件中直接指定——毕竟 Markdown 语法本身并未提供定义此类元数据的标注方式——但您可以通过 `--metadata` 选项或 YAML 格式进行设置（具体方法见下文说明）。

严格来说，生成输出时会用到两种类型的变量：

- 通过 `--metadata` 指定的元数据 (metadata)，
- 通过 `--variable` 指定的模板变量 (variable)。

二者的核心区别在于：元数据可被 Pandoc 及其过滤器（即在最终格式化前处理输入内容的脚本）识别和处理，而模板变量仅作用于模板系统。若您通过 `--metadata` 标签或元数据头文件设置某个变量，该变量同样可供模板系统调用，因此通常只需使用元数据即可。虽然最终输出可能存在细微差异（因为过滤器可能对元数据进行特殊处理而忽略普通变量），但坚持使用单一类型选项显然更简便。除非有充分理由，否则建议统一采用元数据设置方式。

7.1　读取 YAML 元数据

读取 Pandoc 标记文件中的 YAML 元数据可以通过使用 Pandoc 提供的命令行参数或 API 来实现。以下是一个示例命令行使用方式：

```
pandoc --metadata-file=metadata.yaml input.md -o output.html
```

上述命令中，metadata.yaml 是包含 YAML 元数据的文件，input.md 是待转换的标记文件，output.html 是转换后的输出文件。

在实际应用中，读取 Pandoc 标记文件中的 YAML 元数据可以用于自动化文档处理、生成静态网页、构建电子书等场景。通过提取元数据，我们可以根据文档属性进行个性化处理，如根据作者生成不同样式的页面，根据日期进行归档等。

Pandoc 命令中的几个相关选项：

`--metadata-file=FILE` Read metadata from the supplied YAML (or JSON) file.

`-M KEY[=VAL], --metadata=KEY[:VAL]` Set the metadata field KEY to the value VAL.

`-V KEY[=VAL], --variable=KEY[:VAL]` Set the template variable KEY to the string value VAL when rendering the document in standalone mode.

7.2 Pandoc 对元数据的解析

Pandoc 将其解析为 Markdown 格式[1]。我们可以执行命令 `pandoc -t native input.md` 来查看 Pandoc 的解析结果。

- 如果将 `CJKmainfont: 方正楷体_GBK` 写入 metadata.yaml 时，会被解析成 `方正楷体\_GBK`，导致 LaTeX 编译出错。
- 在 `header-includes` 加入 LaTeX 代码时，有可能会解析成 Markdown 文本而发生错误。
- 能够正确以原始代码形式插入的是：`--include-in-header`、`--include-before-body` 和 `--include-after-body`。（见通用写入器选项）

示例：

1. 文件 1 含有下面的 LaTeX 代码：

```
---
header-includes: |
  \renewcommand*{\thefootnote}{(\alph{footnote})}
  \newcommand{\smalltext}[1]{\small#1}
  \usepackage{etoolbox}
```

[1] https://github.com/jgm/pandoc/issues/2139

```
  \pretocmd{\section}{\clearpage}{}{}
  \preto{\section}{\setcounter{footnote}{0}}
  \apptocmd{\toprule}{\rowcolor{cyan!40}}{}{}
---

hello, world
```

Pandoc 解析结果:

```
Pandoc
  Meta
    { unMeta =
        fromList
          [ ( "header-includes"
            , MetaBlocks
                [ RawBlock
                    (Format "tex")
                    "\\renewcommand*{\\thefootnote}{(\\alph{footnote
                       })}\n\\newcommand{\\smalltext}[1]{\\small#1}\
                       n\\usepackage{etoolbox}\n\\pretocmd{\\section
                       }{\\clearpage}{}{}\n\\preto{\\section}{\\
                       setcounter{footnote}{0}} \n\\apptocmd{\\
                       toprule}{\\rowcolor{cyan!40}}{}{}"
                ]
            )
          ]
    }
  [ Para [ Str "hello," , Space , Str "world" ] ]
```

2. 文件 2 含有下面的 LaTeX 代码:

```
---
header-includes: |
  \usepackage{titlesec}

  \titleformat{\chapter}[display]
    {\vspace*{\fill}\centering\Huge\bfseries}
    {第 \thechapter 章}
    {20pt}
    {\centering}
    [\vspace*{\fill}\clearpage]
---

hello, world
```

```
Pandoc
  Meta
    { unMeta =
```

```
        fromList
          [ ( "header-includes"
            , MetaBlocks
                [ RawBlock (Format "tex") "\\usepackage{titlesec}"
                , Para
                    [ Str "\\titleformat{\\chapter}[display]"
                    , SoftBreak
                    , Str "{"
                    , RawInline (Format "tex") "\\vspace*{\\fill}"
                    , RawInline (Format "tex") "\\centering"
                    , RawInline (Format "tex") "\\Huge"
                    , RawInline (Format "tex") "\\bfseries"
                    , Str "}"
                    , SoftBreak
                    , Str "{\31532"
                    , Space
                    , RawInline (Format "tex") "\\thechapter "
                    , Str "\31456}"
                    , SoftBreak
                    , Str "{20pt}"
                    , SoftBreak
                    , Str "{"
                    , RawInline (Format "tex") "\\centering"
                    , Str "}"
                    , SoftBreak
                    , Str "["
                    , RawInline (Format "tex") "\\vspace*{\\fill}"
                    , RawInline (Format "tex") "\\clearpage"
                    , Str "]"
                    ]
                ]
            )
          ]
  }
[ Para [ Str "hello," , Space , Str "world" ] ]
```

后者 LaTeX 代码被解析成字符串文本，编译时将出现错误。

7.3 --include-in-header 与 header-includes

Pandoc 的 `--include-in-header` 命令行选项允许将原始内容直接包含到生成输出文档的头部区域。此内容可以是目标文档格式头部中有效的任何内容，例如：

- LaTeX 前导代码：为 PDF 输出添加自定义命令、包或设置。
- HTML `<head>` 元素：包括样式表（`<link>`）、脚本（`<script>`）或元标签。
- 其他特定于格式的头部元素：根据输出格式，此选项允许直接注入将出现在文档头部的内容。

该选项通常通过指定包含内容的文件名或直接以字符串形式提供内容来使用：

1. 使用文件：

```
pandoc input.md --include-in-header=header.tex -o output.pdf
```

在此示例中，`header.tex` 的内容将被插入到生成的 `output.pdf` 的 LaTeX 前导代码中。

2. 直接提供内容：

```
pandoc input.md --include-in-header='<style>body { color: blue; }</
    style>' -o output.html
```

在此示例中，CSS 样式定义被直接插入到 `output.html` 的 `<head>` 部分。

Pandoc `--include-in-header` 选项直接修改使用的模板，允许对文档头部内容进行精细控制。

- 如果同时存在 `header-includes` 元数据变量或在过滤器中写入 `header-includes` 变量（见过滤器 *Filters*，则该元数据将被 `--include-in-header` 文件内容覆盖，见：
  - https://github.com/jgm/pandoc/issues/3139
  - https://github.com/jgm/pandoc/issues/3138
- 由于 Pandoc 会自动设置一些 `header-includes` 变量（见自动设置的变量），建议使用 `--include-in-header` 时先包含这部分代码，见https://github.com/arcasen/markdown-include/blob/main/preamble.tex。

7.4 命令行选项、元数据块和元数据文件的优先顺序

在 Pandoc 中，元数据（metadata）的优先级遵循以下规则（从高到低）：

1. 命令行参数 (`--metadata` 或 `-M` 选项)
    - 最高优先级，会覆盖其他来源的同名元数据，例如：`pandoc -M title="New Title"`
2. YAML 元数据块 (位于文档头部)
    - 中等优先级，会被命令行参数覆盖，但会覆盖外部元数据文件
    - 例如：

```
---
title: Document Title
author: John Doe
---
```

3. 外部元数据文件 (`--metadata-file` 或 `-d` 选项)
    - 最低优先级，会被以上两者覆盖
    - 例如：`pandoc --metadata-file=meta.yaml`

特殊说明：

- 如果同时使用多个来源定义同一个变量，Pandoc 会按照上述优先级采用值
- YAML 元数据块中可以包含复杂结构（如列表、嵌套对象），而命令行参数只适合简单键值对
- 可以通过 `--variable` (`-V`) 定义的变量属于 LaTeX 模板变量系统，与元数据系统不同但可能有交互

建议实践：

- 将通用元数据放在外部 YAML 文件中
- 文档特定的元数据放在文档头部的 YAML 块中
- 需要临时覆盖时使用命令行参数

7.5 `--defaults`/`-d` 与 `--metadata-file`

Pandoc 的 `-d`（`--defaults`）和 `--metadata-file` 选项都用于从外部文件加载配置，但它们的用途和功能有显著区别：

1. `-d` / `--defaults`（默认配置文件）的用途：定义 Pandoc 转换的全局默认选项（如输入/输出格式、模板、变量、过滤器等）。

2. `--metadata-file`（元数据文件）的用途：仅用于定义文档的元数据（如标题、作者、日期等），不影响转换行为。

表 7.1: `--defaults`/`-d` 与 `--metadata-file` 关键区别对比

特性	`-d` / `--defaults`	`--metadata-file`
影响范围	控制整个转换流程（格式、模板等）	仅设置文档元数据
文件内容	Pandoc 选项 + 元数据	仅元数据
优先级	低于命令行选项	与文档内元数据合并
常用字段示例	`from`, `to`, `filters`	`title`, `author`, `date`

示例：结合使用两者

```
pandoc -d defaults.yaml --metadata-file meta.yaml input.md -o output
    .pdf
```

- `defaults.yaml` 定义转换规则（如 PDF 模板、字体）。
- `meta.yaml` 定义文档内容相关的元数据（如标题、作者）。

第八章　使用模板 Templates

当使用 `-s/--standalone` 选项时，Pandoc 会使用模板添加独立文档所需的头部和尾部内容。要查看使用的默认模板[1]，只需输入：

```
pandoc -D <FORMAT>
```

其中 `FORMAT` 是输出格式的名称。可以使用 `--template` 选项指定自定义模板。

输出格式	模板名称
pdf	default.latex
docx	default.openxml
html	default.html

`docx`、`odt` 和 `pptx` 输出还可以通过 `--reference-doc` 选项进行自定义。使用参考文档调整文档样式；使用模板处理变量插值、自定义元数据的呈现、目录位置、样板文本等。

模板包含变量,允许在文件的任意位置插入任意信息。变量可通过 `-V/--variable` 选项在命令行设置。如果变量未设置，Pandoc 会在文档的元数据中查找键，元数据可通过 YAML 元数据块或 `-M/--metadata` 选项设置。此外，Pandoc 会为某些变量赋予默认值[2]。

如果使用自定义模板，可能需要随着 Pandoc 的更新而修订模板。建议跟踪默认模板的变化，并相应修改自定义模板。

[1] https://github.com/jgm/pandoc-templates

[2] 为利用 Pandoc 这些默认值，最好通过修改 default.latex 或 eisvogel.latex 来自定义模板。例如：当 Markdown 文件中含有图片时，Pandoc 会设置 `graphics: true`，从而包含 `\usepackage{graphicx}` 等命令。

8.1 模板语法

8.1.1 注释

从 `$--` 到行末的任何内容都被视为注释，将从输出中省略。

8.1.2 分隔符

在模板中标记变量和控制结构时，可使用 `$...$` 或 `${...}` 作为分隔符。同一模板中可混合使用这两种样式，但每个分隔符的开头和结尾必须匹配。开头分隔符后可跟一个或多个空格或制表符，这些将被忽略。结尾分隔符前也可有空格或制表符，同样会被忽略。

要在文档中包含字面 `$`，请使用 `$$`。

8.1.3 插值变量

插值变量的占位符是一个由匹配分隔符包围的变量名。变量名必须以字母开头，可包含字母、数字、`_`、`-` 和 `.`。关键字 `it`、`if`、`else`、`endif`、`for`、`sep` 和 `endfor` 不可用作变量名。示例：

```
$foo$
$foo.bar.baz$
$foo_bar.baz-bim$
$ foo $
${foo}
${foo.bar.baz}
${foo_bar.baz-bim}
${ foo }
```

带点的变量名用于访问结构化变量值。例如，`employee.salary` 将返回 `employee` 字段值中 `salary` 字段的值。

- 如果变量值是简单值，将按原样呈现（注意，不会进行转义；假设调用程序会为输出格式适当转义字符串）。
- 如果变量值是列表，则将连接所有值。
- 如果变量值是映射，则呈现字符串 `true`。
- 其他所有值将呈现为空字符串。

8.1.4 条件语句

条件语句以 `if(variable)`（由匹配分隔符包围）开始[3]，以 `endif`（由匹配分隔符包围）结束。可选择包含 `else`（由匹配分隔符包围）。如果变量值为真，则使用 `if` 部分；否则使用 `else` 部分（如果存在）。以下值视为真：

- 任何映射
- 包含至少一个真值的数组
- 任何非空字符串
- 布尔值 `True`

在 YAML 元数据（或通过 `-M/--metadata` 在命令行指定的元数据）中，未加引号的 `true` 和 `false` 将被解释为布尔值。但通过 `-V/--variable` 在命令行指定的变量始终为字符串值。因此，`if(foo)` 条件在 `-V foo=false` 时会被触发，但在 `-M foo=false` 时不会触发。

示例：

```
$if(foo)$bar$endif$

$if(foo)$
  $foo$
$endif$

$if(foo)$
part one
$else$
part two
$endif$

${if(foo)}bar${endif}

${if(foo)}
  ${foo}
${endif}

${if(foo)}
${ foo.bar }
${else}
no foo!
```

[3]Pandoc 只支持对变量值进行 true/false 判断，并不支持变量/常量之间进行比较运算，见：https://github.com/jgm/pandoc/issues/1057。

```
${endif}
```

关键字 `elseif` 可用于简化复杂的嵌套条件：

```
$if(foo)$
XXX
$elseif(bar)$
YYY
$else$
ZZZ
$endif$
```

8.1.5 循环

循环以 `for(variable)`（由匹配分隔符包围）开始，以 `endfor`（由匹配分隔符包围）结束。

- 如果变量是数组，循环内的内容将重复执行，变量依次设置为数组的每个值，并连接结果。
- 如果变量是映射，循环内的内容将设置为该映射。
- 如果关联变量的值不是数组或映射，将对其值执行单次迭代。

示例：

```
$for(foo)$$foo$$sep$, $endfor$

$for(foo)$
  - $foo.last$, $foo.first$
$endfor$

${ for(foo.bar) }
  - ${ foo.bar.last }, ${ foo.bar.first }
${ endfor }

$for(mymap)$
$it.name$: $it.office$
$endfor$
```

可通过 `sep`（由匹配分隔符包围）指定连续值之间的分隔符。`sep` 和 `endfor` 之间的内容为分隔符。

```
${ for(foo) }${ foo }${ sep }, ${ endfor }
```

在循环中，可使用特殊代词关键字 `it` 代替变量名。

```
${ for(foo.bar) }
  - ${ it.last }, ${ it.first }
${ endfor }
```

8.1.6 部分模板

部分模板（存储在不同文件中的子模板）可通过部分模板名称后跟 () 包含，例如：

```
${ styles() }
```

部分模板将在主模板所在目录中查找。如果文件名没有扩展名，将假设与主模板的扩展名相同。调用部分模板时，也可使用包含文件扩展名的完整名称：

```
${ styles.html() }
```

（如果在模板目录中未找到部分模板，且模板路径为相对路径，还将在用户数据目录的 templates 子目录中查找。）

部分模板可通过冒号应用于变量：

```
${ date:fancy() }

${ articles:bibentry() }
```

如果 articles 是数组，将对其值迭代，依次应用 bibentry() 部分模板。因此，上述第二个示例等价于：

```
${ for(articles) }
${ it:bibentry() }
${ endfor }
```

迭代部分模板时，必须使用代词关键字 it。在上述示例中，bibentry 部分模板应包含 it.title（依此类推），而不是 articles.title。

包含的部分模板会省略末尾换行符。

部分模板可包含其他部分模板。

数组值之间的分隔符可在变量名或部分模板后立即用方括号指定：

```
${months[, ]}
```

```
${articles:bibentry()[; ]}
```

此处的分隔符为字面值（与显式 `for` 循环中的 `sep` 不同），不能包含插值变量或其他模板指令。

8.1.7 嵌套

为确保内容“嵌套”（即后续行缩进），使用 `^` 指令：

```
$item.number$  $^$$item.description$ ($item.price$)
```

在此例中，如果 `item.description` 包含多行，所有行将缩进以与首行对齐：

```
00123  A fine bottle of 18-year old
       Oban whiskey. ($148)
```

要将多行嵌套到同一级别，请在模板中将它们与 `^` 指令对齐。例如：

```
$item.number$  $^$$item.description$ ($item.price$)
               (Available til $item.sellby$.)
```

将生成：

```
00123  A fine bottle of 18-year old
       Oban whiskey. ($148)
       (Available til March 30, 2020.)
```

如果变量单独出现在一行，前有空格且同一行后无其他文本或指令，且变量值包含多行，则会自动嵌套。

8.1.8 可断行空格

通常，模板本身的空格（区别于插值变量的值）不可断行，但可通过使用 `~` 关键字（以另一个 `~` 结束）使部分模板中的空格可断行。

```
$~$This long line may break if the document is rendered
with a short line length.$~$
```

8.1.9 管道

管道用于转换变量或部分模板的值。管道通过变量名（或部分模板）与管道名称之间的斜杠 (/) 指定。示例：

```
$for(name)$
$name/uppercase$
$endfor$

$for(metadata/pairs)$
- $it.key$: $it.value$
$endfor$

$employee:name()/uppercase$
```

管道可链式使用：

```
$for(employees/pairs)$
$it.key/alpha/uppercase$. $it.name$
$endfor$
```

某些管道接受参数：

```
|---------------------|------------|
$for(employee)$
$it.name.first/uppercase/left 20 "| "$$it.name.salary/right 10 " | "
    " |"$
$endfor$
|---------------------|------------|
```

当前预定义的管道包括：

`pairs`：将映射或数组转换为包含 `key` 和 `value` 字段的映射数组。如果原始值是数组，`key` 将为数组索引，从 1 开始。

`uppercase`：将文本转换为大写。

`lowercase`：将文本转换为小写。

`length`：返回值的长度：文本值的字符数，映射或数组的元素数。

`reverse`：反转文本值或数组，对其他值无影响。

`first`：返回非空数组的第一个值；否则返回原始值。

`last`：返回非空数组的最后一个值；否则返回原始值。

`rest`：返回非空数组除第一个值外的所有值；否则返回原始值。

`allbutlast`：返回非空数组除最后一个值外的所有值；否则返回原始值。

`chomp`：移除尾随换行符（和可断行空格）。

`nowrap`：禁用可断行空格的换行。

`alpha`：将可解读为整数的文本值转换为小写字母 `a..z`（模 26）。可用于从数组索引生成字母编号。链式使用 `uppercase` 可生成大写字母。

`roman`：将可解读为整数的文本值转换为小写罗马数字。可用于从数组索引生成罗马数字编号。链式使用 `uppercase` 可生成大写罗马数字。

`left n "leftborder" "rightborder"`：将文本值渲染为宽度 `n` 的块，左对齐，带可选的左右边界。对其他值无影响。可用于表格内容对齐。宽度为正整数，表示字符数。边界为双引号中的字符串；字面 `"` 和 `\` 字符必须反斜杠转义。

`right n "leftborder" "rightborder"`：将文本值渲染为宽度 `n` 的块，右对齐，对其他值无影响。

`center n "leftborder" "rightborder"`：将文本值渲染为宽度 `n` 的块，居中对齐，对其他值无影响。

8.2 变量

8.2.1 元数据变量

`title`, `author`, `date`：用于标识文档的基本信息。可以通过 LaTeX 和 ConTeXt 将这些信息嵌入到 PDF 元数据中。可以通过 Pandoc 标题块设置，标题块支持多个作者，也可以通过 YAML 元数据块设置：

```
---
author:
- 亚里士多德
- 彼得·阿贝拉尔
...
```

如果仅需设置 PDF 或 HTML 的元数据，而不在文档中包含标题块，可以设置 `title-meta`、`author-meta` 和 `date-meta` 变量。（默认情况下，这些变量会根据 `title`、`author` 和 `date` 自动设置。）HTML 页面标题由 `pagetitle` 设置，默认与 `title` 相同。

`subtitle`：文档副标题，包含在 HTML、EPUB、LaTeX、ConTeXt 和 docx 文档中。

`abstract`：文档摘要，包含在 HTML、LaTeX、ConTeXt、AsciiDoc 和 docx 文档中。

`abstract-title`：摘要标题，目前仅用于 HTML、EPUB 和 docx。会根据 `lang` 自动设置为本地化值，但可以手动覆盖。

`keywords`：关键词列表，包含在 HTML、PDF、ODT、pptx、docx 和 AsciiDoc 元数据中；与 `author` 类似，可重复设置。

`subject`：文档主题，包含在 ODT、PDF、docx、EPUB 和 pptx 元数据中。

`description`：文档描述，包含在 ODT、docx 和 pptx 元数据中。某些应用程序将其显示为 “注释” 元数据。

`category`：文档类别，包含在 docx 和 pptx 元数据中。

此外，任何未包含在 ODT、docx 或 pptx 元数据中的根级字符串元数据将作为自定义属性添加。例如，以下 YAML 元数据块：

```
---
title: '这是标题'
subtitle: "这是副标题"
author:
- 作者一
- 作者二
description: |
    这是一个较长的描述。

    它包含两段文字。
...
```

在转换为 docx、ODT 或 pptx 时，`title`、`author` 和 `description` 将作为标准文档属性，`subtitle` 将作为自定义属性。

8.2.2 语言变量

`lang`：使用 IETF 语言标签（遵循 BCP 47 标准）标识文档的主要语言，例如 `en` 或 `en-GB`。可使用语言子标签查询工具验证这些标签。影响大多数格式，并在使用 LaTeX（通过 babel 和 polyglossia）或 ConTeXt 生成 PDF 时控制连字符。

使用带有 `lang` 属性的原生 Pandoc Div 和 Span 切换语言：

```
---
lang: en-GB
...

Text in the main document language (British English).

::: {lang=fr-CA}
> Cette citation est écrite en français canadien.
:::

More text in English. ['Zitat auf Deutsch.']{lang=de}
```

`dir`：基本书写方向，值为 `rtl`（从右到左）或 `ltr`（从左到右）。

对于双向文档，可以使用带有 `dir` 属性（值为 `rtl` 或 `ltr`）的原生 Pandoc Span 和 Div 来覆盖某些输出格式的默认方向。如果最终渲染器（如生成 HTML 时的浏览器）支持 Unicode 双向算法，可能不需要此操作。

在使用 LaTeX 处理双向文档时，仅 `xelatex` 引擎完全支持（使用 `--pdf-engine=xelatex`）。

8.2.3 HTML 变量

`document-css`：启用 `styles.html` 模板中大部分 CSS 的包含（可通过 `Pandoc --print-default-data-file=templates/styles.html` 查看）。默认情况下，除非使用 `--css`，此变量为 `true`。可通过 `Pandoc -M document-css=false` 禁用。

`mainfont`：设置 HTML 元素的 CSS `font-family` 属性。

`fontsize`：设置基础 CSS `font-size`，通常设置为例如 `20px`，但也接受 `pt`（在大多数浏览器中，12pt = 16px）。

`fontcolor`：设置 HTML 元素的 CSS `color` 属性。

`linkcolor`：设置所有链接的 CSS `color` 属性。

`monofont`：设置代码元素的 CSS `font-family` 属性。

`monobackgroundcolor`：设置代码元素的 CSS `background-color` 属性并添加额外内边距。

`linestretch`：设置 HTML 元素的 CSS `line-height` 属性，推荐使用无单位

值。

`maxwidth`：设置 CSS `max-width` 属性（默认值为 36em）。

`backgroundcolor`：设置 HTML 元素的 CSS `background-color` 属性。

`margin-left`, `margin-right`, `margin-top`, `margin-bottom`: 设置 body 元素的相应 CSS 内边距属性。

若需为单个文档覆盖或扩展 CSS，可包含例如：

```
---
header-includes: |
  <style>
  blockquote {
    font-style: italic;
  }
  tr.even {
    background-color: #f0f0f0;
  }
  td, th {
    padding: 0.5em 2em 0.5em 0.5em;
  }
  tbody {
    border-bottom: none;
  }
  </style>
---
```

8.2.4 HTML 数学变量

`classoption`：当使用 `--katex` 时，可以通过 YAML 元数据或 `-M classoption=fleqn` 设置数学公式左对齐显示。

8.2.5 HTML 幻灯片变量

这些变量影响使用 Pandoc 生成幻灯片时的 HTML 输出。

`institute`：作者隶属机构：支持多个作者时可以是列表。

`revealjs-url`：reveal.js 文档的基础 URL（默认为 https://unpkg.com/reveal.js@%5E5 ）。

`s5-url`：S5 文档的基础 URL（默认为 `s5/default`）。

`slidy-url`：Slidy 文档的基础 URL（默认为 https://www.w3.org/Talks/Tools/Slidy2 ）。

`slideous-url`：Slideous 文档的基础 URL（默认为 `slideous`）。

`title-slide-attributes`：reveal.js 幻灯片标题页的额外属性。参见 reveal.js、Beamer 和 pptx 中的背景示例[4]。

所有 reveal.js 配置选项[5]均可用作变量。若需关闭 reveal.js 中默认开启的布尔标志，使用 `0`。

8.2.6 Beamer 幻灯片变量

这些变量更改使用 Beamer 生成的 PDF 幻灯片的外观。

`aspectratio`：幻灯片宽高比（43 表示 4:3 [默认]，169 表示 16:9，1610 表示 16:10，149 表示 14:9，141 表示 1.41:1，54 表示 5:4，32 表示 3:2）。

`beameroption`：通过 `\setbeameroption{}` 添加额外的 Beamer 选项。

`institute`：作者隶属机构：支持多个作者时可以是列表。

`logo`：幻灯片 logo 图片。

`navigation`：控制导航符号（默认为空，即无导航符号；其他有效值为 `frame`、`vertical` 和 `horizontal`）。

`section-titles`：启用新部分的 “标题页”（默认为 `true`）。

`theme`, `colortheme`, `fonttheme`, `innertheme`, `outertheme`：Beamer 主题。

`themeoptions`, `colorthemeoptions`, `fontthemeoptions`, `innerthemeoptions`, `outerthemeoptions`：LaTeX Beamer 主题的选项（列表）。

`titlegraphic`：标题页图片：可以是列表。

`titlegraphicoptions`：标题页图片的选项。

`shorttitle`, `shortsubtitle`, `shortauthor`, `shortinstitute`, `shortdate`：某些 Beamer 主题使用标题、副标题、作者、机构、日期的短版本。

[4] https://pandoc.org/MANUAL.html#background-in-reveal.js-Beamer-and-pptx

[5] https://revealjs.com/config/

8.2.7 PowerPoint 变量

这些变量控制幻灯片中不易通过模板控制的视觉效果。

`monofont`：用于代码的字体。

8.2.8 LaTeX 变量

Pandoc 在使用 LaTeX 引擎生成 PDF 时使用这些变量。

8.2.8.1 布局

`block-headings`：使 `\paragraph` 和 `\subparagraph`（第四级和第五级标题，或在 book 类中为第五级和第六级）独立显示，而非嵌入式；需要进一步格式化以区分 `\subsubsection`（第三级或第四级标题）。与其使用此选项，KOMA-Script 可以更广泛地调整标题：

```
---
documentclass: scrartcl
header-includes: |
  \RedeclareSectionCommand[
    beforeskip=-10pt plus -2pt minus -1pt,
    afterskip=1sp plus -1sp minus 1sp,
    font=\normalfont\itshape]{paragraph}
  \RedeclareSectionCommand[
    beforeskip=-10pt plus -2pt minus -1pt,
    afterskip=1sp plus -1sp minus 1sp,
    font=\normalfont\scshape,
    indent=0pt]{subparagraph}
...
```

`classoption`：文档类的选项，例如 `oneside`；可重复设置多个选项：

```
---
classoption:
- twocolumn
- landscape
...
```

`documentclass`：文档类：通常为标准类 `article`、`book` 和 `report`；KOMA-Script 类似的是 `scrartcl`、`scrbook` 和 `scrreprt`（默认较小边距）；或 `memoir`。

`geometry`：geometry 包的选项，例如 `margin=1in`；可重复设置多个选项：

```
---
geometry:
- top=30mm
- left=20mm
- heightrounded
...
```

`hyperrefoptions`：hyperref 包的选项，例如 `linktoc=all`；可重复设置多个选项：

```
---
hyperrefoptions:
- linktoc=all
- pdfwindowui
- pdfpagemode=FullScreen
...
```

`indent`：如果为 `true`，Pandoc 将使用文档类的缩进设置（否则默认 LaTeX 模板会移除缩进并在段落间添加间距）。

`linestretch`：使用 setspace 包调整行间距，例如 `1.25`、`1.5`。

`margin-left`, `margin-right`, `margin-top`, `margin-bottom`：如果未使用 geometry 包，则设置边距（否则 geometry 会覆盖这些设置）。

`pagestyle`：控制 `\pagestyle{}`：默认 article 类支持 `plain`（默认）、`empty`（无页眉页脚）、`headings`（页眉包含章节标题）。

`papersize`：纸张大小，例如 `letter`、`a4`。

`secnumdepth`:章节编号深度(配合 `--number-sections` 选项或 `numbersections` 变量)。

`beamerarticle`：从 Beamer 幻灯片生成文章。**注意**：若设置此变量，必须指定 Beamer 写入器，但使用默认 LaTeX 模板，例如：`Pandoc -Vbeamerarticle -t beamer --template default.latex`。

`handout`：生成 Beamer 幻灯片的讲义版本（将叠加内容压缩为单页）。

`csquotes`：加载 csquotes 包并对引文使用 `\enquote` 或 `\enquote*`。

`csquotesoptions`：csquotes 包的选项（可重复设置多个选项）。

`babeloptions`：传递给 babel 包的选项（可重复设置多个选项）。如果主要语言不是使用拉丁或西里尔字母的欧洲语言或越南语，默认设置为 `provide=*`。大多数用

户无需调整默认设置。

8.2.8.2 字体

`fontenc`：通过 fontenc 包指定字体编码（配合 pdflatex）；默认值为 `T1`（参见 LaTeX 字体编码指南）。

`fontfamily`：配合 `pdflatex` 使用的字体包：TeX Live 包含许多选项，详见 LaTeX 字体目录。默认值为 Latin Modern。

`fontfamilyoptions`：作为 `fontfamily` 的包的选项；可重复设置多个选项。例如，通过 `libertinus` 宏包[6]使用 Libertine 字体并启用比例小写（旧式）数字：

```
---
fontfamily: libertinus
fontfamilyoptions:
- osf
- p
...
```

`fontsize`：正文字体大小。标准类支持 10pt、11pt 和 12pt。若需其他大小，设置 `documentclass` 为 KOMA-Script 类，如 `scrartcl` 或 `scrbook`。

`mainfont`, `sansfont`, `monofont`, `mathfont`, `CJKmainfont`, `CJKsansfont`, `CJKmonofont`：用于 `xelatex` 或 `lualatex` 的字体族：可使用系统字体名称，配合 `fontspec` 包。`CJKmainfont` 在 `xelatex` 中使用 `xecjk` 包，在 `lualatex` 中使用 `luatexja` 包。

`mainfontoptions`, `sansfontoptions`, `monofontoptions`, `mathfontoptions`, `CJKoptions`, `luatexjapresetoptions`：在 `xelatex` 和 `lualatex` 中用于 `mainfont`、`sansfont`、`monofont`、`mathfont`、`CJKmainfont` 的选项，支持 `fontspec` 包的任何选项；可重复设置多个选项。例如，使用 TeX Gyre 版本的 Palatino 字体并启用小写数字：

```
---
mainfont: TeX Gyre Pagella
mainfontoptions:
- Numbers=Lowercase
- Numbers=Proportional
...
```

[6] https://ctan.org/pkg/libertinus

`mainfontfallback`, `sansfontfallback`, `monofontfallback`：如果在 `mainfont`、`sansfont` 或 `monofont` 中找不到字形，则尝试使用的字体列表。字体名称后必须跟冒号，可选附加选项，例如：

```
---
mainfontfallback:
  - "FreeSans:"
  - "NotoColorEmoji:mode=harf"
...
```

字体回退目前仅支持 `lualatex`。

`babelfonts`：Babel 语言名称（例如 `chinese`）与对应字体的映射：

```
---
babelfonts:
  chinese-hant: "Noto Serif CJK TC"
  russian: "Noto Serif"
...
```

`microtypeoptions`: 传递给 `microtype` 包的选项。

8.2.8.3 链接

`colorlinks`：为链接文本添加颜色；如果设置了 `linkcolor`、`filecolor`、`citecolor`、`urlcolor` 或 `toccolor`，则自动启用。

`boxlinks`：在链接周围添加可见框（如果设置了 `colorlinks`，则无效）。

`linkcolor`, `filecolor`, `citecolor`, `urlcolor`, `toccolor`
分别为内部链接、外部链接、引用链接、URL 链接和目录链接设置颜色：支持 `xcolor` 包的选项，包括 `dvipsnames`、`svgnames` 和 `x11names` 颜色列表。

`links-as-notes`：将链接作为脚注打印。

`urlstyle`：URL 的字体样式（例如 `tt`、`rm`、`sf`，默认为 `same`）。

8.2.8.4 前置内容

`lof`, `lot`：包含图表列表和表格列表（也可通过 `--lof`/`--list-of-figures`、`--lot`/`--list-of-tables` 设置）。

`thanks`：文档标题后致谢脚注的内容。

`toc`：包含目录（也可通过 `--toc/--table-of-contents` 设置）。

`toc-depth`：目录中包含的章节级别。

8.2.8.5 BibLaTeX 参考文献

这些变量在使用 BibLaTeX 渲染引用时生效。

`biblatexoptions`：`biblatex` 的选项列表。

`biblio-style`：参考文献样式，配合 `--natbib` 和 `--biblatex` 使用。

`biblio-title`：参考文献标题，配合 `--natbib` 和 `--biblatex` 使用。

`bibliography`：用于解析引用的参考文献文件。

`natbiboptions`：natbib 的选项列表。

8.2.9 ConTeXt 变量

Pandoc 在使用 ConTeXt 生成 PDF 时使用这些变量。

`fontsize`：正文字体大小（例如 `10pt`、`12pt`）。

`headertext`, `footertext`：放置在页眉或页脚的文本（参见 ConTeXt 页眉和页脚）；可重复设置最多四次以实现不同位置。

`indenting`：控制段落缩进，例如 `yes,small,next`（参见 ConTeXt 缩进）；可重复设置多个选项。

`interlinespace`：调整行间距，例如 `4ex`（使用 `setupinterlinespace`）；可重复设置多个选项。

`layout`：页面边距和文本排列的选项（参见 ConTeXt 布局）；可重复设置多个选项。

`linkcolor`, `contrastcolor`：页面内外链接的颜色，例如 `red`、`blue`（参见 ConTeXt 颜色）。

`linkstyle`：链接的字体样式，例如 `normal`、`bold`、`slanted`、`boldslanted`、`type`、`cap`、`small`。

`lof`, `lot`：包含图表列表和表格列表。

`mainfont`, `sansfont`, `monofont`, `mathfont`：字体族：使用任何系统字体名称（参见 ConTeXt 字体切换）。

`mainfontfallback`, `sansfontfallback`, `monofontfallback`：如果主字体中缺少字形，则按顺序尝试的字体列表，使用 `\definefallbackfamily` 兼容的字体名称语法。不支持表情符号字体。

`margin-left`, `margin-right`, `margin-top`, `margin-bottom`：如果未使用 `layout`，则设置边距（否则 `layout` 会覆盖这些设置）。

`pagenumbering`：页面编号样式和位置（使用 `setuppagenumbering`）；可重复设置多个选项。

`papersize`：纸张大小，例如 `letter`、`A4`、`landscape`（参见 ConTeXt 纸张设置）；可重复设置多个选项。

`pdfa`：添加生成指定类型 PDF/A 所需的设置，例如 `1a:2005`、`2a`。如果未指定类型（例如通过 `--metadata=pdfa` 或 YAML 中 `pdfa: true` 设置），默认使用 `1b:2005` 以保持向后兼容性。不支持未指定值的 `--variable=pdfa`。生成 PDF/A 需确保 ICC 颜色配置文件可用，且内容及包含文件（如图片）符合标准。ICC 配置文件和输出意图可通过 `pdfaiccprofile` 和 `pdfaintent` 变量指定。参见 ConTeXt PDFA 了解详情。

`pdfaiccprofile`：配合 `pdfa` 使用，指定 PDF 使用的 ICC 配置文件，例如 `default.cmyk`。如果未指定，默认使用 `sRGB.icc`。可重复设置多个配置文件。配置文件需在系统上可用，可从 ConTeXt ICC 配置文件获取。

`pdfaintent`：配合 `pdfa` 使用，指定颜色输出意图，例如 `ISO coated v2 300\letterpercent\space (ECI)`。如果未指定，默认使用 `sRGB IEC61966-2.1`。

`toc`：包含目录（也可通过 `--toc/--table-of-contents` 设置）。

`urlstyle`：无链接文本的 URL 字体样式，例如 `normal`、`bold`、`slanted`、`boldslanted`、`type`、`cap`、`small`。

`whitespace`：段落间距，例如 `none`、`small`（使用 `setupwhitespace`）。

`includesource`：将所有源文档作为 PDF 文件附件包含。

8.2.10 wkhtmltopdf 变量

Pandoc 在使用 wkhtmltopdf 生成 PDF 时使用这些变量。`--css` 选项也会影响输出。

`footer-html`, `header-html`：为页眉和页脚添加信息。

`margin-left`, `margin-right`, `margin-top`, `margin-bottom`：设置页面边距。

`papersize`：设置 PDF 纸张大小。

8.2.11 man 页面变量

`adjusting`：调整文本对齐方式，值为左（`l`）、右（`r`）、居中（`c`）或两端（`b`）。

`footer`：man 页面页脚。

`header`：man 页面页眉。

`section`：man 页面章节编号。

8.2.12 Texinfo 变量

`version`：软件版本（用于标题和标题页）。

`filename`：生成 info 文件的名称（默认为基于 texi 文件名的名称）。

8.2.13 Typst 变量

`template`：使用的 Typst 模板。

`margin`：Typst 文档中定义的边距字段：`x`、`y`、`top`、`bottom`、`left`、`right`。

`papersize`：纸张大小：`a4`、`us-letter` 等。

`mainfont`：主字体的系统字体名称。

`fontsize`：字体大小（例如 `12pt`）。

`section-numbering`：章节编号模式，例如 `1.A.1`。

`page-numbering`：页面编号模式，例如 `1` 或 `i`，或空字符串以省略页面编号。

`columns`：正文列数。

8.2.14 ms 变量

`fontfamily`: 字体: `A` (Avant Garde)、`B` (Bookman)、`C` (Helvetica)、`HN` (Helvetica Narrow)、`P` (Palatino) 或 `T` (Times New Roman)。此设置不影响源代码，源代码始终使用等宽 Courier 显示。这些内置字体字符覆盖范围有限。可使用 Peter Schaffter 提供的 `install-font.sh`[7] 脚本安装额外字体，详见其网站[8]。

`indent`: 段落缩进（例如 `2m`）。

`lineheight`: 行高（例如 `12p`）。

`pointsize`: 字体大小（例如 `10p`）。

8.3 自动设置的变量

Pandoc 根据选项[9]或文档内容自动设置这些变量，用户也可修改。这些变量因输出格式而异，包括:

`body`: 文档正文。

`date-meta`: 转换为 ISO 8601 `YYYY-MM-DD` 格式的日期, 包含在所有基于 HTML 的格式（dzslides、epub、html、html4、html5、revealjs、s5、slideous、slidy）中。支持的日期格式为: `mm/dd/yyyy`、`mm/dd/yy`、`yyyy-mm-dd` (ISO 8601)、`dd MM yyyy` (例如 `02 Apr 2018` 或 `02 April 2018`)、`MM dd, yyyy` (例如 `Apr. 02, 2018` 或 `April 02, 2018`)、`yyyy[mm[dd]]` (例如 `20180402`、`201804` 或 `2018`)。

`header-includes`: 由 `-H/--include-in-header` 指定的内容(可有多个值)。

`include-before`: 由 `-B/--include-before-body` 指定的内容（可有多个值)。

`include-after`: 由 `-A/--include-after-body` 指定的内容 (可有多个值)。

`meta-json`: 文档所有元数据的 JSON 表示，字段值转换为所选输出格式。

`numbersections`: 如果指定了 `-N/--number-sections`，则为非空值。

`sourcefile`, `outputfile`: 命令行中给定的源文件和目标文件名。如果输入来自多个文件，`sourcefile` 可以是列表；如果输入来自标准输入，则为空。以下模板片

[7] https://www.schaffter.ca/mom/bin/install-font.sh

[8] https://www.schaffter.ca/mom/momdoc/appendices.html#steps

[9] https://pandoc.org/MANUAL.html#options

段可区分它们:

```
$if(sourcefile)$
$for(sourcefile)$
$sourcefile$
$endfor$
$else$
(stdin)
$endif$
```

类似地，`outputfile` 如果输出到终端，则为 `-`。

如果需要绝对路径，可使用例如 `$curdir$/$sourcefile$`。

`pdf-engine`：如果通过 `--pdf-engine` 提供 PDF 引擎名称，或请求 PDF 输出时的默认引擎。

`curdir`：运行 Pandoc 的工作目录。

`pandoc-version`：Pandoc 版本。

`toc`：如果指定了 `--toc/--table-of-contents`，则为非空值。

`toc-title`：目录标题（仅适用于 EPUB、HTML、revealjs、opendocument、odt、docx、pptx、Beamer、LaTeX）。

在 docx 和 pptx 中，自定义 `toc-title` 可从元数据中获取，但无法作为变量设置。

在使用 Pandoc 将文档转换为 PDF 时，Pandoc 会根据输入文件的内容、元数据（metadata）以及使用的模板自动生成一些变量。这些变量主要用于模板渲染，特别是在 LaTeX 或其他模板中生成 PDF 时。例如，即使不设置 `header-includes`，Pandoc 也会自动设置一些变量，可以查看生成的 LaTeX 即知。在自动设置的变量中定义了:

- `figurename`
- `tablename`
- `listfigurename`
- `listtablename`
- `lstlistlistingname`
- 等等

如果在文档中重新设置 `header-includes`，将被置于内置的 `header-`

includes 之前，无法修改这些自动设置的变量。可以在 LaTeX 模板的 header-includes 之后增加 header-continue。这样，我们就可以在文档中设置新的数据，如：

```
header-continue: |
  \AtBeginDocument{%
    \renewcommand*\figurename{图}
    \renewcommand*\tablename{表}
    \renewcommand*\lstlistingname{代码}
    \renewcommand*\listfigurename{图表清单}
    \renewcommand*\listtablename{表格清单}
    \renewcommand*\lstlistlistingname{代码清单}
  }
```

第九章　使用样式参考文档

`--reference-doc` 是 Pandoc 中用于指定模板文档（docx、ODT 或 pptx 格式）的选项[1]，可在转换为同类文件时保留模板的样式和格式。尤其适用于需要严格匹配 Word（.docx）、LibreOffice（.odt）或 PowerPoint（.pptx）模板的场景。基本用法：

```
pandoc input.md --reference-doc=reference.docx -o output.docx
```

这个命令会使用 `reference.docx` 中的样式来格式化 `output.docx`。

核心功能：

- 样式继承
  - Pandoc 会沿用参考文档中的样式（标题、列表、表格等）。
  - 适用于需要符合机构/学校模板要求的场景。
- 格式控制
  - 字体、页边距、页面大小、页眉/页脚等布局设置均从模板继承。
- 兼容性
  - 支持.docx、.odt（LibreOffice）和.pptx（演示文稿）。

9.1　参考 docx 文件

为了获得最佳结果，参考的 docx 文件应是使用 Pandoc 生成的 docx 文件的修改版本。参考 docx 文件的内容会被忽略，但其样式表和文档属性（包括边距、页面大小、页眉和页脚）将被用于新的 docx 文件。如果在命令行中未指定参考 docx 文件，Pandoc 将在用户数据目录（参见 `--data-dir` 选项）中查找名为 reference.docx 的文件。

[1] https://pandoc.org/MANUAL.html#option--reference-doc

`--reference-docx` 选项的参数应该是指向 reference.docx 文件的路径[2]。如果你在数据目录（而非工作目录）中放置了一个 reference.docx 文件，并使用 `--reference-docx reference.docx` 命令，程序将无法找到该文件。这不是错误，而是设计如此。当不使用 `--reference-docx` 选项时，用户数据目录中的 reference.docx 文件会覆盖系统默认设置。

要创建自定义 reference.docx 文件，首先获取默认 reference.docx 文件的副本：`pandoc -o custom-reference.docx --print-default-data-file reference.docx` 。然后在 Word 中打开 custom-reference.docx，修改样式并保存文件。为获得最佳结果，除修改 Pandoc 使用的样式外，不要对该文件进行其他更改：

段落样式：

- 普通
- 正文
- 首段
- 紧凑
- 标题
- 副标题
- 作者
- 日期
- 摘要
- 摘要标题
- 参考文献
- 标题 1
- 标题 2
- 标题 3
- 标题 4
- 标题 5
- 标题 6
- 标题 7
- 标题 8
- 标题 9
- 块文本 [用于块引用]
- 脚注块文本 [用于脚注中的块引用]
- 源代码
- 脚注文本
- 定义术语

[2] https://github.com/jgm/pandoc/issues/2965

- 定义
- 说明文字
- 表格说明
- 图片说明
- 图表
- 带说明的图表
- 目录标题

字符样式:

- 默认段落字体
- 逐字字符
- 脚注引用
- 超链接
- 章节编号

表格样式:

- 表格

9.2 参考 pptx 文件

微软 PowerPoint 2013 包含的模板（扩展名为.pptx 或.potx）已知是可用的，大多数基于这些模板衍生的模板也是如此。

具体要求是模板必须包含以下名称的布局（在 PowerPoint 中可见）:

- 标题幻灯片
- 标题和内容
- 节标题
- 双内容
- 比较
- 带说明的内容
- 空白

对于每个名称，将使用找到的第一个具有该名称的布局。如果找不到某个名称的布局，Pandoc 将输出警告，并使用默认参考文档中该名称的布局。（这些布局的使用方式在 PowerPoint 布局选择中有描述。）

最近版本的 MS PowerPoint 包含的所有模板都符合这些标准。（您可以在 “开始” 菜单下点击 “布局” 来检查。）

您还可以修改默认的 reference.pptx：首先运行命令 `pandoc -o custom-reference.pptx --print-default-data-file reference.pptx`，然后在 MS PowerPoint 中修改 custom-reference.pptx（Pandoc 将使用上述列出的布局名称）。

9.3 pptx 分页

在使用 Pandoc 生成 PowerPoint（.pptx）文件时，可以通过在 Markdown 文件中设置标题级别或使用特定的分隔符来实现分页（即创建新的幻灯片）。以下是具体的方法和步骤：

9.3.1 使用标题级别控制分页

Pandoc 在生成 pptx 时，会根据 Markdown 文件中的标题级别（#, ##, ### 等）来决定幻灯片的划分。默认情况下，Pandoc 使用最高级别标题之后有内容的标题级别作为幻灯片标题（slide level）。

示例：

```
# 章节标题
## 第一页标题
这里是第一页的内容。
- 列表项 1
- 列表项 2

## 第二页标题
这里是第二页的内容。
- 列表项 A
- 列表项 B
```

说明：

- # 级别（H1）通常用作章节标题（title slide），表示一个部分的开始。
- ## 级别（H2）被用作每页幻灯片的标题。
- 如果你希望更低级别的标题（如 ###）作为幻灯片标题，可以通过 –slide-level 参数指定，例如：

  ```
  pandoc -t pptx --slide-level=3 input.md -o output.pptx
  ```

  这会将 ### 作为幻灯片标题级别。

命令：

```
pandoc -t pptx input.md -o output.pptx
```

每个 ## 标题会生成一个新的幻灯片，标题内容为 ## 的文本，后面跟着该标题下的内容。

- 如果文档中没有足够的标题级别，Pandoc 会自动选择最高标题级别作为幻灯片标题。
- 如果你不想用标题来分页，可以设置 `--slide-level=0`，然后使用其他方式（如分隔符）来控制分页。

9.3.2 使用分隔符手动分页

Pandoc 支持使用水平线（---）作为幻灯片的分隔符来手动分页。这种方式适用于不需要严格标题结构的场景。

示例：

```
# 章节标题
## 第一页标题
这里是第一页的内容。
- 列表项 1
- 列表项 2

---

## 第二页标题
这里是第二页的内容。
- 列表项 A
- 列表项 B
```

说明：

- --- 表示一个新的幻灯片开始。
- 在 PowerPoint 输出中，Pandoc 不需要额外的 `-s`（standalone）参数，因为 pptx 文件本身就是独立的。

命令：

```
pandoc -t pptx input.md -o output.pptx
```

每个 `---` 会创建一个新的幻灯片，`---` 之前的标题（如 `# 演示标题`）会成为演示文稿的标题页。

第十章　过滤器 Filters

10.1　Pandoc 的抽象语法树 AST

10.1.1　Haskell 表示

Pandoc 的 `-t native` 选项将输入文档转换为 Pandoc 的内部表示格式, 即 Haskell 数据结构。具体来说，`-t native` 的输出是 Pandoc 的 Native AST（抽象语法树），本质上是 Pandoc 的文档模型（Abstract Syntax Tree, AST）的序列化形式，用 Haskell 的语法表达。这种格式主要用于调试或开发 Pandoc 相关工具，方便开发者查看 Pandoc 如何解析文档的内部结构。

示例：测试文档 lorem-ipsum.md 的内容如下：

```
# Lorem ipsum

*Lorem ipsum* dolor sit amet
```

执行 `pandoc lorem-ipsum.md -t native` 输出：

```
Pandoc
  Meta { unMeta = fromList [] }
  [ Header
      1
      ( "lorem-ipsum" , [] , [] )
      [ Str "Lorem" , Space , Str "ipsum" ]
  , Para
      [ Emph [ Str "Lorem" , Space , Str "ipsum" ]
      , Space
      , Str "dolor"
      , Space
      , Str "sit"
      , Space
      , Str "amet"
      ]
```

```
]
```

10.1.2 JSON 表示

Pandoc 的 `-t json` 选项将输入文档转换为 JSON 格式的输出。具体来说，这是 Pandoc 的内部文档模型（Abstract Syntax Tree, AST）的 JSON 表示。输出的格式是 JSON（JavaScript Object Notation），一种通用的、语言无关的数据交换格式。Pandoc 使用 JSON 来序列化其内部 AST，描述文档的结构（如标题、段落、强调等）。JSON 输出表示了 Pandoc 解析文档后的结构化数据，包含元数据（metadata）和文档内容的层级结构。这种格式常用于程序化处理 Pandoc 的文档数据，例如与其他工具集成、自动化处理或调试 Pandoc 的解析结果。

执行 `pandoc lorem-ipsum.md -t json | python3 -m json.tool` 输出：

```
{
    "pandoc-api-version": [
        1,
        23,
        1
    ],
    "meta": {},
    "blocks": [
        {
            "t": "Header",
            "c": [
                1,
                [
                    "lorem-ipsum",
                    [],
                    []
                ],
                [
                    {
                        "t": "Str",
                        "c": "Lorem"
                    },
                    {
                        "t": "Space"
                    },
                    {
                        "t": "Str",
                        "c": "ipsum"
                    }
```

```
                ]
            ]
        },
        {
            "t": "Para",
            "c": [
                {
                    "t": "Emph",
                    "c": [
                        {
                            "t": "Str",
                            "c": "Lorem"
                        },
                        {
                            "t": "Space"
                        },
                        {
                            "t": "Str",
                            "c": "ipsum"
                        }
                    ]
                },
                {
                    "t": "Space"
                },
                {
                    "t": "Str",
                    "c": "dolor"
                },
                {
                    "t": "Space"
                },
                {
                    "t": "Str",
                    "c": "sit"
                },
                {
                    "t": "Space"
                },
                {
                    "t": "Str",
                    "c": "amet"
                }
            ]
        }
    ]
}
```

10.2 使用过滤器

Pandoc 为用户提供了一个接口，用于编写程序（称为过滤器）[1]，这些程序作用于 Pandoc 的抽象语法树（AST）。

Pandoc 由一组读取器和写入器组成。在将文档从一种格式转换为另一种格式时，读取器会将文本解析为 Pandoc 的文档中间表示 —— 即 “抽象语法树” 或 AST —— 然后由写入器将其转换为目标格式。

“过滤器” 是一个在读取器和写入器之间修改 AST 的程序。

```
INPUT --reader--> AST --filter(s)--> AST --writer--> OUTPUT
```

Pandoc 支持两种类型的过滤器：

Lua 过滤器 使用 Lua 语言定义对 Pandoc AST 的转换。

JSON 过滤器 是从标准输入读取并向标准输出写入的管道，处理和生成 Pandoc AST 的 JSON 表示。

文档的 JSON 表示（representation）可以看作一棵树。文档包含段落，段落包含单词和空格，某些单词被强调，依此类推。Pandoc 过滤器包通常会遍历这种树结构，并对你提供的每个节点应用一个函数。这个函数可以保持局部树结构不变，也可以返回一个修改后的树。过滤器的输出将是带有你修改的输入树。你可以使用下面的命令查看文件的 JSON 表示（使用 json.tool 将 JSON 内容格式化输出）：

```
pandoc input.md --to json | python3 -m json.tool > output.json
```

传递给 `--filter`/`-F` 参数的文件应为可执行文件。然而，如果文件的可执行位未设置，Pandoc 会尝试根据文件扩展名猜测合适的解释器[2]。

文件扩展名	解释器
.py	python
.hs	runhaskell
.pl	perl
.rb	ruby
.php	php
.js	node
.r	Rscript

[1] https://pandoc.org/filters.html

[2] https://pandoc.org/filters.html#supported-interpreters

示例：

```
pandoc input.md --filter /path/to/filtername
```

上面的实例中使用了过滤器的绝对路径或相对路径，我们也可以将过滤器放入 Pandoc 用户文件目录：

```
mkdir -p ~/.local/share/pandoc/filters
cp -p /path/to/filtername ~/.local/share/pandoc/filters
```

在 Linux/Unix 系统中，`cp` 命令的 `-p` 选项的作用是保留源文件的元数据，包括以下内容：

1. 文件权限：保留源文件的权限设置（读、写、执行等）。
2. 时间戳：保留源文件的访问时间（atime）、修改时间（mtime）和创建时间（ctime，如果系统支持）。
3. 文件所有者和所属组：尽可能保留源文件的所有者和组信息（需要有相应权限，通常需要超级用户权限）。

我们可以在任务地方以如下方式调用过滤器：

```
pandoc input.md --filter filtername
```

10.3 使用 Panflute

Pandoc 过滤器是一个程序，主要功能是：

1. 接收 Pandoc 文档的 JSON 表示
2. 修改文档
3. 输出修改后的 JSON 表示

Panflute 简化了这个过程，让你可以用 Python 轻松操作文档。

先运行以下命令安装 Panflute：

```
$ pip3 install panflute
```

默认情况下，Pandoc 调用系统的 Python 解释器（如：macOS 是 python2），而上述过滤器需要调用 python3 解释器。解决方案如下（这里特意去掉了过滤器文件的后缀.py。）：
方案 1：建立虚拟环境，如：

```
mkvirtualenv pandoc
pip install panflute
workon pandoc
```

方案 2：设置过滤器为可执行文件
在 uppercase 开始加入：

```
#!/usr/bin/env python3
```

然后，设置过滤器为可执行文件：

```
chmod +x uppercase
```

下面的实例中将将 Markdown 文本的字母转换为大写字母：

```
import panflute as pf

def upper(elem, doc):
  if isinstance(elem, pf.Str):
    elem.text = elem.text.upper()

if __name__ == "__main__":
  pf.run_filter(upper)
```

```
Hello, World

你好，世界
```

```
pandoc helloworld.md --filter ./uppercase
<p>HELLO, WORLD</p>
<p>你好，世界</p>
```

10.4 过滤器实例：启用 LaTeX 环境

下面是一个用 Panflute 编写的过滤器，可以处理 Pandoc 的 Div，将其转换成对应类名的 LaTeX 环境。以下是具体的渲染规则和输出示例：

1. 第一个类（class）：如果存在类，会使用第一个类名相应的 LaTeX 环境名称。
2. 其它类：其它类将转化成 LaTeX 环境的参数。
3. id：通常被转换为 LaTeX 的 `\label` 命令，用于交叉引用。
4. 属性：键值对（如 `key=value`）也将转化成 LaTeX 环境的参数。

该过滤器支持 LaTeX 模板中的如下形式的环境：

```
\begin{env_name}[env_options]\label{env_label}
<content>
\end{env_name}
```

代码如下：

```
#!/usr/bin/env python3

import panflute as pf


def action(elem, doc):
    if isinstance(elem, pf.Div) and doc.format == 'latex':
        # 获取 div 的属性
        classes = elem.classes
        identifier = elem.identifier
        attributes = elem.attributes

        # 如果 div 有 class，则转换为对应的 LaTeX 环境
        if classes:
            # 获取第一个 class 作为环境名
            env_name = elem.classes[0]

            # 处理 id（转换为 \label）
            label = f'\\label{{{identifier}}}' if elem. identifier
               else ''

            # 处理其它类名（如：.cls1 .cls2）
            cls_str = ','.join(
                f'{cls}' for cls in classes[1:]) if classes[1:] else
                    ''

            # 处理自定义属性（如：key1=val1 key2= val2）
            attr_str = ','.join([
                    f'{key}={value}' for key, value in attributes.
                       items()
                ]) if attributes else ''

            # 生成 LaTeX 环境的参数
            # 用逗号连接有效字符串，并过滤掉空字符串（None 或 ""）
```

```
            opt_str = ','.join([s for s in [cls_str, attr_str] if s
                ])
            opt_str = f'[{opt_str}]' if opt_str else ''

            # 创建 LaTeX 环境
            begin_env = pf.RawBlock(
                f'\\begin{{{env_name}}}' + opt_str + label, format='
                    latex')
            end_env = pf.RawBlock(f'\\end{{{env_name}}}', format='
                latex')

            # 如果有 label，将其插入到 div 内容开头
            content = list(elem.content)

            # 返回新的元素列表：开始环境 + 内容 + 结束环境
            return [begin_env] + content + [end_env]


if __name__ == '__main__':
    pf.run_filter(action, doc=None)
```

与参考文献（见文献引用 *Citations*）有冲突，编译时应先使用 `--filter latex-environment`，后使用 `--citeproc`；否则，无法定义 `\citeproc` 等命令。

10.4.1 定理环境

通过这个过滤器，我们可以在 Markdown 中使用定理环境：

```
下面是韦达定理（Vieta's Formulas）的[描述](#vieta)和证明。

::: {.theorem #vieta}

如果 $x_1,x_2$ 是一元二次方程 $ax^2 + bx + c = 0$ 的两个根，则有：

$$ x_1 + x_2 = -\dfrac{b}{a} $$

$$ x_1 \cdot x_2 = \dfrac{c}{a} $$

:::

::: proof

因为 $x_1,x_2$ 是一元二次方程 $ax^2 + bx + c = 0$ 的两个根，于是有：
```

```
$$ ax^2 + bx + c = a(x-x_1)(x-x_2) $$

根据乘法原理展开右式，比较等号两边的各项系数可得韦达定理的叙述。

:::
```

下面是韦达定理（Vieta's Formulas）的描述和证明。

定理 10.1 如果 x_1, x_2 是一元二次方程 $ax^2 + bx + c = 0$ 的两个根，则有：

$$x_1 + x_2 = -\frac{b}{a}$$

$$x_1 \cdot x_2 = \frac{c}{a}$$

证明 因为 x_1, x_2 是一元二次方程 $ax^2 + bx + c = 0$ 的两个根，于是有：

$$ax^2 + bx + c = a(x - x_1)(x - x_2)$$

根据乘法原理展开右式，比较等号两边的各项系数可得韦达定理的叙述。

10.4.2 awesomebox

通过这个过滤器，我们可以在 Markdown 中使用 awesomebox[3] 环境：

```
::: note

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nam aliquet
    libero quis lectus elementum fermentum.

:::

::: tip
```

[3] https://ctan.org/pkg/awesomebox

```
Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nam aliquet
    libero quis lectus elementum fermentum.

:::

::: warning

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nam aliquet
    libero quis lectus elementum fermentum.

:::

::: caution

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nam aliquet
    libero quis lectus elementum fermentum.

:::

::: important

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nam aliquet
    libero quis lectus elementum fermentum.

:::
```

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nam aliquet libero quis lectus elementum fermentum.

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nam aliquet libero quis lectus elementum fermentum.

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nam aliquet libero quis lectus elementum fermentum.

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nam aliquet libero quis lectus elementum fermentum.

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nam aliquet libero quis lectus elementum fermentum.

awesomebox 内部不能使用脚注。

10.4.3 `messagebox` 环境

示例：在 Stenciler 模板中预定义了 `messagebox` 环境，我们可以按如下方法使用。

messagebox 可以嵌套使用。

```
::: {.messagebox title="Lorem ipsum"}

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nam aliquet
    libero quis lectus elementum fermentum.

:::
```

Lorem ipsum

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nam aliquet libero quis lectus elementum fermentum.

10.5 过滤器实例：条纹表格 Striped Table

下面是一个用 Panflute 编写的过滤器，可以处理生成条纹表格（Striped Table）。代码如下：

```python
#!/usr/bin/env python3

# 替换表格标题中的 .striped 字符串，并重新以 Pandoc 方式解析
# 使其能够被 pandoc-crossref 调用

import re
from functools import reduce
import panflute as pf


def action(elem, doc):
    # 检查元素是否为表格且有标题
    if isinstance(elem, pf.Table) and elem.caption:
        # 将标题内容转换为纯文本
        caption_text = pf.stringify(elem.caption)
        # 判断是否包含 .striped
        if '.striped' in caption_text:
            replacements = [
                (r'\{\s+', '{'),           # 去掉 { 后面的空格
                (r'\s+\}', '}'),           # 去掉 } 前面的空格
                (r'\s*\.striped\s*', ''),  # 确定 .striped 子字符串
                (r'\{\}', ''),             # 确定 {} 子字符串
                (r'^\s+', ''),             # 去掉前导空格
                (r'\s+$', ''),             # 确定尾部空格
            ]
            # reduce 将 caption_text 作为初始值，依次对每对 (pattern, repl) 调用
            # re.sub()，每次的结果作为下一次的输入
            new_caption_text = reduce(lambda s, pr: re.sub(pr[0], pr[1], s),
                                      replacements,
                                      caption_text)

            # 如果替换后标题不为空，重新解析为 Pandoc AST
            if new_caption_text:
                # 使用 convert_text 将新标题字符串解析为 AST
                parsed_doc = pf.convert_text(new_caption_text,
                                             input_format='markdown',
                                             standalone=False)
                # 创建新的 Caption 元素，提取解析后的内联内容
                new_caption = pf.Caption(
```

```
                    *parsed_doc) if parsed_doc else pf.Caption()
                elem.caption = new_caption
            else:
                # 如果标题为空，移除标题
                elem.caption = None

            # 如果输出格式是 LaTeX，添加 LaTeX 代码
            if doc.format == 'latex':
                return [pf.RawBlock(r'{\apptocmd{\toprule}{\rowcolor{cyan!40}}{}{}',
                                    format='latex'),
                        pf.RawBlock(
                            r'\rowcolors{1}{white}{cyan!15}', format='latex'),
                        elem,
                        pf.RawBlock(r'}', format='latex')]
            elif doc.format == "html":
                # HTML 方式：为表格添加 CSS 类
                elem.classes = ["striped-table"]
                return elem


def finalize(doc):
    # HTML 输出时插入 CSS 样式
    if doc.format == "html":
        css = """<style>
      table.striped-table tr:nth-child(even) { background: #f0f0f0; }
    </style>"""
        doc.metadata["header-includes"] = pf.MetaBlocks(
            pf.RawBlock(css, format="html"))


if __name__ == '__main__':
    pf.run_filter(action, finalize=finalize, doc=None)
```

使用方法：在表格标题添加 `.striped`。

> 如何与 pandoc-crossref 一起使用时，请调用 striped-table。

示例：

```
| Type                           | Description               | Example Use
                                                 |
```

```
| ------------------------ | --------------------- |
  ------------------------------------------------ |
| String                   | Single line of text   | Clause
  number e.g. "Section 1.2"                        |
| Text                     | Multiple lines of text | Description
  of requirement                                   |
| Date                     | Date                  | Date of
  object creation. Format defined in options. |
| Enumeration type, single | Defined by types      | Dropdown
  selection from a formatted list                  |
| Enumeration type, multiple | Defined by types    | Multiple
  choice selections from a limited list            |

: Striped Table {#tbl:striped-tbl .striped}

| Type                     | Description           | Example Use
                                                   |
| ------------------------ | --------------------- |
  ------------------------------------------------ |
| String                   | Single line of text   | Clause
  number e.g. "Section 1.2"                        |
| Text                     | Multiple lines of text | Description
  of requirement                                   |
| Date                     | Date                  | Date of
  object creation. Format defined in options. |
| Enumeration type, single | Defined by types      | Dropdown
  selection from a formatted list                  |
| Enumeration type, multiple | Defined by types    | Multiple
  choice selections from a limited list            |

: Table without Strip {#tbl:simple-tbl}

条纹表格见 [@tbl:striped-tbl] 普通表格见 [@tbl:simple-tbl]
```

表 10.2: Striped Table

Type	Description	Example Use
String	Single line of text	Clause number e.g. “Section 1.2”
Text	Multiple lines of text	Description of requirement
Date	Date	Date of object creation. Format defined in options.

Type	Description	Example Use
Enumeration type, single	Defined by types	Dropdown selection from a formatted list
Enumeration type, multiple	Defined by types	Multiple choice selections from a limited list

表 10.3: Table without Strip

Type	Description	Example Use
String	Single line of text	Clause number e.g. “Section 1.2”
Text	Multiple lines of text	Description of requirement
Date	Date	Date of object creation. Format defined in options.
Enumeration type, single	Defined by types	Dropdown selection from a formatted list
Enumeration type, multiple	Defined by types	Multiple choice selections from a limited list

条纹表格见表 10.2 普通表格见表 10.3

10.6 过滤器实例：设置文字颜色

下面是一个用 Panflute 编写的过滤器，可以处理两种颜色标记格式，支持以 # 开头的 HTML 十六进制颜色代码，并同时兼容 HTML 和 PDF/LaTeX 输出：

1. 行内（Span）颜色

```
[文字]{color=颜色}
```

2. 块级（Div）颜色

```
:::{color=颜色}
文字内容
:::
```

代码如下:

```
#!/usr/bin/env python3

import panflute as pf


def action(elem, doc):
    # 处理行内颜色 [文字]{color=颜色}
    if isinstance(elem, pf.Span) and 'color' in elem.attributes:
        color = elem.attributes['color']

        if doc.format in ('html', 'html5'):
            # HTML 输出 - 直接使用颜色值
            elem.attributes['style'] = f'color: {color};'

        elif doc.format == 'latex':
            # LaTeX 输出 - 处理不同颜色格式
            if color.startswith('#'):
                # 十六进制颜色
                elem.content = ([
                    pf.RawInline(f'\\textcolor[HTML]{{{color[1:]}}}{{',
                                 format='latex')]
                                + list(elem.content)
                                + [pf.RawInline('}', format='latex')])
            else:
                # 标准颜色名称
                elem.content = ([
                    pf.RawInline(f'\\textcolor{{{color}}}{{',
                                 format='latex')]
                                + list(elem.content)
                                + [pf.RawInline('}', format='latex')])

        del elem.attributes['color']
        return elem

    # 处理块级颜色 ::: {color=颜色} 文字内容 :::
    elif isinstance(elem, pf.Div) and 'color' in elem.attributes:
        color = elem.attributes['color']

        if doc.format in ('html', 'html5'):
            # HTML 输出
            elem.attributes['style'] = f'color: {color};'

        elif doc.format == 'latex':
```

```
            # LaTeX 输出
            if color.startswith('#'):
                # 十六进制颜色
                elem.content = ([
                    pf.RawBlock(f'{{\\color[HTML]{{{color[1:]}}}}',
                                format='latex')]
                                + list(elem.content)
                                + [pf.RawBlock('}', format='latex')
                                   ])
            else:
                # 标准颜色名称
                elem.content = ([
                    pf.RawBlock(f'{{\\color{{{color}}}}',
                                format='latex')]
                                + list(elem.content)
                                + [pf.RawBlock('}', format='latex')
                                   ])

        del elem.attributes['color']
        return elem


if __name__ == '__main__':
    pf.run_filter(action, doc=None)
```

使用说明：

1. 将上述代码保存为 `text-color`
2. 使用 Pandoc 转换时添加过滤器：

```
# 转换为 HTML
pandoc input.md -o output.html --filter text-color

# 转换为 PDF
pandoc input.md -o output.pdf --filter text-color
```

功能特点：

1. 支持 HTML 十六进制颜色：
   - 可以处理 `[文字]{color=#FF0000}` 这样的十六进制颜色代码
   - 也可以处理标准颜色名称如 `[文字]{color=red}`
2. 自动适应输出格式：
   - HTML 输出：使用 `style="color: #FF0000;"`
   - LaTeX 输出：自动生成 `\definecolor` 命令和 `\textcolor` 环境
3. 块级和行内颜色支持：

- 行内：`[红色文字]{color=#FF0000}`
- 块级：

```
::: {color=#00FF00}
这里是绿色文字块
:::
```

注意事项：

1. 确保 LaTeX 安装了 `xcolor` 包
2. 对于 PDF 输出，复杂的颜色组合可能需要额外的 LaTeX 包
3. 颜色名称和十六进制值需要是有效的 CSS/LateX 颜色值

示例：

```
对 Pandoc Span & Div 属性的扩展，为文本着色：

- Span

  [标准颜色]{color=purple}和[HTML 颜色]{color=#6c3483}

- Div

  :::{color=teal}
  使用标准颜色的文本块
  :::

  :::{color=#27ae60}
  使用 HTML 颜色的文本块
  :::
```

对 Pandoc Span & Div 属性的扩展，为文本着色：

- Span

  标准颜色和HTML 颜色

- Div

  使用标准颜色的文本块

  使用 HTML 颜色的文本块

10.7 支持 docx、HTML 和 PDF 格式输出

以下是支持 docx、HTML 和 PDF 输出的完整过滤器，能够处理以 # 开头的十六进制颜色代码：

```
#!/usr/bin/env python3

import panflute as pf


def action(elem, doc):
    # 处理行内颜色 [文字]{color=颜色}
    if isinstance(elem, pf.Span) and 'color' in elem.attributes:
        color = elem.attributes['color']

        if doc.format in ('html', 'html5'):
            # HTML 输出
            elem.attributes['style'] = f'color: {color};'

        elif doc.format == 'latex':
            # LaTeX/PDF 输出
            if color.startswith('#'):
                # 十六进制颜色
                elem.content = ([
                    pf.RawInline(f'\\textcolor[HTML]{{{color
                       [1:]}}}{{',
                                 format='latex')]
                                + list(elem.content)
                                + [pf.RawInline('}', format='latex')
                                   ])
            else:
                # 标准颜色名称
                elem.content = ([
                    pf.RawInline(f'\\textcolor{{{color}}}{{',
                                 format='latex')]
                                + list(elem.content)
                                + [pf.RawInline('}', format='latex')
                                   ])

        elif doc.format == 'docx':
            # DOCX 输出 - 使用 OpenXML 属性
            elem.attributes['custom-style'] = 'ColoredText'
            # 将颜色信息存储在临时属性中，后续处理
            elem.attributes['_color'] = color

        del elem.attributes['color']
        return elem
```

```
    # 处理块级颜色 ::: {color=颜色} 文字内容 :::
    elif isinstance(elem, pf.Div) and 'color' in elem.attributes:
        color = elem.attributes['color']

        if doc.format in ('html', 'html5'):
            elem.attributes['style'] = f'color: {color};'

        elif doc.format == 'latex':
            if color.startswith('#'):
                # 十六进制颜色
                elem.content = ([
                    pf.RawBlock(f'{{\\color[HTML]{{{color[1:]}}}',
                                format='latex')]
                                + list(elem.content)
                                + [pf.RawBlock('}', format='latex')
                                   ])
            else:
                # 标准颜色名称
                elem.content = ([
                    pf.RawBlock(f'{{\\color{{{color}}}',
                                format='latex')]
                                + list(elem.content)
                                + [pf.RawBlock('}', format='latex')
                                   ])

        elif doc.format == 'docx':
            # DOCX 输出 - 使用 pf.Div 的 custom-style
            elem.attributes['custom-style'] = 'ColoredBlock'
            # 存储颜色信息
            elem.attributes['_color'] = color

        del elem.attributes['color']
        return elem


if __name__ == '__main__':
    pf.run_filter(action, doc=None)
```

使用说明:

1. 将上述代码保存为 `text-color-plus`
2. 准备一个 Word 参考文档 `custom-reference.docx` 包含以下样式:
    - `ColoredText` - 用于行内彩色文本
    - `ColoredBlock` - 用于彩色文本块

3. 使用 Pandoc 转换时添加过滤器：

```
# 转换为 HTML
pandoc input.md -o output.html --filter text-color-plus

# 转换为 PDF
pandoc input.md -o output.pdf --filter text-color-plus

# 转换为 docx
pandoc input.md -o output.docx --filter text-color-plus
            --reference-doc=custom-reference.docx
```

注意事项：

1. docx 输出需要参考文档中的预定义样式
2. 对于 docx 输出，颜色信息存储在 `_color` 属性中，实际渲染需要 Word 应用这些样式
3. 更高级的 docx 处理可能需要直接操作 OpenXML，这超出了 Panflute 的基本功能
4. 确保在所有输出格式中测试颜色显示效果，因为不同格式对颜色的支持可能有所不同

使用步骤：

1. 创建参考 Word 文档：
    - 新建一个 Word 文档
    - 创建两个样式：
        - `ColoredText`：字符样式，可以设置任意颜色（实际颜色将由过滤器覆盖）
        - `ColoredBlock`：段落样式，可以设置任意颜色
    - 保存为 `custom-reference.docx`
2. 修改过滤器（可选）：
    - 如果需要更精细的 docx 样式控制，可以扩展过滤器以生成更复杂的 OpenXML 属性
3. 颜色映射（可选）：
    - 可以添加颜色名称到十六进制值的映射表，确保 docx 中的颜色一致性

完整解决方案文件结构：

```
.
├── text-color-plus             # 过滤器脚本
├── custom-reference.docx       # Word 参考文档
├── input.md                    # 示例输入文件
└── Makefile                    # 构建脚本示例
```

命令:

```
#!/bin/bash
# 生成 HTML
pandoc input.md -o output.html --filter ./text-color-plus

# 生成 PDF
pandoc input.md -o output.pdf --filter ./text-color-plus

# 生成 docx
pandoc input.md -o output.docx --filter ./text-color-plus
   --reference-doc=custom-reference.docx
```

第十一章　幻灯片展示

您可以使用 Pandoc 生成基于 HTML + JavaScript 的幻灯片演示文稿，可通过网页浏览器查看。有五种方式可以实现：使用 S5、DZSlides、Slidy、Slideous 或 reveal.js。您还可以使用 LaTeX Beamer 生成 PDF 格式的幻灯片，或生成 Microsoft PowerPoint 格式的幻灯片。

以下是一个简单的幻灯片展示的 Markdown 源文件，名为 habits.txt：

```
% Habits
% John Doe
% March 22, 2005

# In the morning

## Getting up

- Turn off alarm
- Get out of bed

## Breakfast

- Eat eggs
- Drink coffee

# In the evening

## Dinner

- Eat spaghetti
- Drink wine

------------------

![picture of spaghetti](chapters/images/spaghetti.jpg)

## Going to sleep
```

```
- Get in bed
- Count sheep
```

要生成 HTML/JavaScript 幻灯片，只需输入：

```
pandoc -t FORMAT -s habits.txt -o habits.html
```

其中 FORMAT 可以是 s5、slidy、slideous、dzslides 或 revealjs。

对于 Slidy、Slideous、reveal.js 和 S5，使用 `-s/--standalone` 选项生成的文件的 JavaScript 和 CSS 文件链接，默认假设这些文件位于相对路径 s5/default（S5）、slideous（Slideous）、reveal.js（reveal.js）或 Slidy 的 w3.org 网站（Slidy）。（这些路径可以通过设置 `slidy-url`、`slideous-url`、`revealjs-url` 或 `s5-url` 变量更改；见上文 HTML 幻灯片变量。）对于 DZSlides，默认情况下，JavaScript 和 CSS（相对较短）会包含在文件中。

对于所有 HTML 幻灯片格式，可以使用 `--self-contained` 选项生成一个包含所有展示所需数据的单一文件，包括链接的脚本、样式表、图片和视频。

要使用 Beamer 生成 PDF 幻灯片，输入：

```
pandoc -t Beamer habits.txt -o habits.pdf
```

reveal.js 幻灯片还可以通过浏览器打印到文件转换为 PDF。

要生成 PowerPoint 幻灯片，输入：

```
pandoc habits.txt -o habits.pptx
```

11.1 幻灯片结构

默认情况下，幻灯片级别是文档中最高标题级别，且该标题后紧跟内容而非另一个标题。在上例中，1 级标题总是后跟 2 级标题，2 级标题后跟内容，因此幻灯片级别为 2。可以使用 `--slide-level` 选项覆盖此默认设置。

文档按以下规则分割为幻灯片：

- 水平线始终开始一个新幻灯片。
- 幻灯片级别的标题始终开始一个新幻灯片。

- 低于幻灯片级别的标题在幻灯片内创建子标题。(在 Beamer 中, 会创建一个 “块”。如果标题带有 `example` 类，将使用 `exampleblock` 环境；如果带有 `alert` 类，将使用 `alertblock`；否则使用普通块。)
- 高于幻灯片级别的标题创建 “标题幻灯片”，仅包含章节标题，用于将幻灯片展示分段。HTML 幻灯片中，这些标题下的非幻灯片内容将包含在标题幻灯片中；Beamer 中，则包含在后续同标题的幻灯片中。
- 如果文档包含标题块，会自动生成标题页。(在 Beamer 中，可通过注释默认模板中的某些行禁用此功能。)

这些规则支持多种幻灯片展示风格。如果您不关心幻灯片的章节和子章节结构，可以仅使用 1 级标题（此时幻灯片级别为 1），或设置 `--slide-level=0`。

在 reveal.js 幻灯片中，如果幻灯片级别为 2，将生成二维布局，1 级标题水平构建，2 级标题垂直构建。建议不要在 reveal.js 中使用更深的嵌套，除非设置 `--slide-level=0`（此时 reveal.js 生成一维布局，仅将水平线视为幻灯片分隔)。

11.2 PowerPoint 布局选择

创建幻灯片时，pptx 编写器根据幻灯片内容从预定义布局中选择:

- **标题幻灯片**：用于初始幻灯片，基于元数据字段（日期、作者、标题）生成和填充（如果存在)。
- **章节标题**：用于 pandoc 称为 “标题幻灯片” 的幻灯片，即以高于幻灯片级别的标题开始的幻灯片。
- **双列内容**：用于双列幻灯片，即包含带有 `columns` 类的 div，且其中至少有两个带有 `column` 类的 div。
- **比较**：用于双列幻灯片中至少一列包含文本后跟非文本（例如图片或表格）的场景，替代 “双列内容”。
- **带标题的内容**：用于非双列幻灯片，包含文本后跟非文本（例如图片或表格)。
- **空白**：用于仅包含空白内容的幻灯片，例如仅包含演讲者笔记或仅包含不间断空格的幻灯片。
- **标题和内容**：用于不符合其他布局条件的幻灯片。

这些布局从 pandoc 包含的默认 pptx 参考文档中选择，除非使用 `--reference-doc` 指定了其他参考文档。

11.3 增量列表

默认情况下，列表一次性显示所有内容。如果希望列表逐项显示，使用 `-i` 选项。如果希望特定列表偏离默认设置，可将其放入带有 `incremental` 或 `nonincremental` 类的 div 块中。例如，使用围栏式 div 语法，以下内容无论文档默认设置如何都将逐项显示：

```
::: incremental

- Eat spaghetti
- Drink wine

:::
```

或：

```
::: nonincremental

- Eat spaghetti
- Drink wine

:::
```

虽然推荐使用 `incremental` 和 `nonincremental` div 设置逐个列表的增量显示，但也支持一种较旧的方法：将列表放入引用块会偏离文档默认设置（即不使用 `-i` 选项时逐项显示，使用 `-i` 选项时一次性显示）：

```
> - Eat spaghetti
> - Drink wine
```

两种方法都允许在同一文档中混合增量和非增量列表。

如果想包含引用块中的列表，可以通过将列表放入围栏式 div 绕过此行为，使其不直接作为引用块的子节点：

```
> ::: wrapper
> - a
> - list in a quote
> :::
```

11.4 插入暂停

您可以通过包含一个包含三个点的段落（点之间用空格分隔）在幻灯片内添加“暂停”：

```
# Slide with a pause

content before the pause

. . .

content after the pause
```

此功能尚未在 PowerPoint 输出中实现。

11.5 幻灯片样式

您可以通过在用户数据目录（见上文 `--data-dir`）的 `$DATADIR/s5/default`（S5）、`$DATADIR/slidy`（Slidy）或 `$DATADIR/slideous`（Slideous）中放置自定义 CSS 文件更改 HTML 幻灯片的样式。原始文件可在 pandoc 的系统数据目录（通常为 `$CABALDIR/pandoc-VERSION/s5/default`）中找到。如果用户数据目录中没有找到文件，pandoc 将在系统数据目录中查找。

对于 dzslides，CSS 包含在 HTML 文件本身中，可在其中修改。

所有 reveal.js 配置选项都可以通过变量设置。例如，可以通过设置 `theme` 变量使用主题：

```
-V theme=moon
```

或者可以使用 `--css` 选项指定自定义样式表。

要为 Beamer 幻灯片设置样式，可以使用 `-V` 选项指定主题、颜色主题、字体主题、内部主题和外部主题：

```
pandoc -t Beamer habits.txt -V theme:Warsaw -o habits.pdf
```

在 HTML 幻灯片格式中，标题属性将转换为幻灯片属性（在 `<div>` 或 `<section>` 上），允许您为单个幻灯片设置样式。在 Beamer 中，某些标题类和属性被识别为框架选项，并作为选项传递给框架（见下文 Beamer 的框架属性）。

11.6 演讲者笔记

reveal.js、PowerPoint (pptx) 和 Beamer 输出支持演讲者笔记。您可以在 Markdown 文档中添加笔记，如下所示：

```
::: notes

This is my note.

- It can contain Markdown
- like this list

:::
```

在 reveal.js 中，按 `s` 键查看演示时可显示笔记窗口。在 PowerPoint 中，演讲者笔记将在讲义和演示者视图中正常可用。

其他幻灯片格式尚不支持笔记，但笔记不会显示在幻灯片本身上。

11.7 列布局

要将内容并排放置在列中，可以使用带有 `columns` 类的原生 div 容器，其中包含两个或更多带有 `column` 类和 `width` 属性的 div 容器：

```
:::::::::::::: {.columns}
::: {.column width="40%"}
contents...
:::
::: {.column width="60%"}
contents...
:::
::::::::::::::
```

当前不支持在 PowerPoint 中指定列宽。

11.8 Beamer 中的额外列属性

带有 columns 和 column 类的 div 容器可以选择性地具有 align 属性。columns 类还可以选择性地具有 totalwidth 属性或 onlytextwidth 类：

```
:::::::::::::: {.columns align=center totalwidth=8em}
::: {.column width="40%"}
contents...
:::
::: {.column width="60%" align=bottom}
contents...
:::
::::::::::::::
```

columns 和 column 上的 align 属性可以取值 top、top-baseline、center 和 bottom，用于垂直对齐列，默认值为 top。

totalwidth 属性将列的总宽度限制为指定值：

```
:::::::::::::: {.columns align=top .onlytextwidth}
::: {.column width="40%" align=center}
contents...
:::
::: {.column width="60%"}
contents...
:::
::::::::::::::
```

onlytextwidth 类将 totalwidth 设置为 \textwidth。

详情请参阅《Beamer 用户指南》第 12.7 节。

11.9 Beamer 中的框架属性

有时需要在 Beamer 的框架中添加 LaTeX [fragile] 选项（例如，使用 minted 环境时）。可以通过在引入幻灯片的标题中添加 fragile 类强制实现：

```
# Fragile slide {.fragile}
```

《Beamer 用户指南》第 8.1 节中描述的所有其他框架属性也都可以使用：allowdisplaybreaks、allowframebreaks、b、c、s、t、environment、label、plain、shrink、standout、noframenumbering、squeeze。特别推荐使用 allowframebreaks，尤其是在参考文献中，因为它允许在内容溢出框架时创建多个幻灯片：

```
# References {.allowframebreaks}
```

此外，frameoptions 属性可用于将任意框架选项传递给 Beamer 幻灯片：

```
# Heading {frameoptions="squeeze,shrink,customoption=foobar"}
```

11.10 reveal.js、Beamer 和 pptx 中的背景

可以在自包含的 reveal.js 幻灯片、Beamer 幻灯片和 pptx 幻灯片中添加背景图片。

1. 所有幻灯片（Beamer、reveal.js、pptx）

在 Beamer 和 reveal.js 中，可以通过 YAML 元数据块或命令行变量使用 background-image 配置选项，在每页幻灯片上设置相同的背景图片。

对于 reveal.js，background-image 将作为 parallaxBackgroundImage 使用（见下文）。

对于 pptx，可以使用 --reference-doc，在相关布局上设置背景图片。

2. parallaxBackgroundImage（reveal.js）

对于 reveal.js，还可以使用 reveal.js 原生选项 parallaxBackgroundImage，生成视差滚动背景。必须同时设置 parallaxBackgroundSize，并可以选择设置 parallaxBackgroundHorizontal 和 parallaxBackgroundVertical 来配置滚动行为。有关这些选项的含义，请参阅 reveal.js 文档。

在 reveal.js 的概览模式中，parallaxBackgroundImage 仅在第一页幻灯片上显示。

3. 单个幻灯片（reveal.js、pptx）

要在特定 reveal.js 或 pptx 幻灯片上设置背景图片，请在幻灯片的第一个幻灯片级别标题上添加 {background-image="/path/to/image"}（标题甚至可以为空）。

由于 HTML 编写器会传递未知属性，reveal.js 的其他背景设置也适用于单个幻灯片，包括 background-size、background-repeat、background-color、

`transition` 和 `transition-speed`。（`data-` 前缀会自动添加。）

为与 reveal.js 保持一致，pptx 也支持 `data-background-image`——如果未找到 `background-image`，将检查 `data-background-image`。

4. 标题幻灯片（reveal.js、pptx）

要在 reveal.js 的自动生成标题幻灯片上添加背景图片，请在 YAML 元数据块中使用 `title-slide-attributes` 变量，该变量必须包含属性名称和值的映射。（注意，此处需要 `data-` 前缀，因为它不会自动添加。）

对于 pptx，请传递一个在“标题幻灯片”布局上设置了背景图片的 `--reference-doc`。

示例（reveal.js）：

```
---
title: My Slide Show
parallaxBackgroundImage: /path/to/my/background_image.png
title-slide-attributes:
    data-background-image: /path/to/title_image.png
    data-background-size: contain
---

## Slide One

Slide 1 has background_image.png as its background.

## {background-image="/path/to/special_image.jpg"}

Slide 2 has a special image for its background, even though the
   heading has no content.
```

第十二章　使用 Markmap 制作思维导图

12.1　什么是 Markmap？

Markmap[1] 是一个将 Markdown 文本转换为交互式思维导图的工具，简单易用，适合整理笔记、规划项目或可视化复杂信息。Markmap 是一个 JavaScript 库，能够解析 Markdown 文本的层级结构（如标题、列表等），并将其渲染为交互式思维导图。它支持：

- Markdown 语法（链接、粗体、斜体、代码块等）
- 数学公式（通过 KaTeX）
- 动态交互（缩放、拖动、折叠/展开节点）
- 导出为 HTML 或 SVG 文件

12.2　使用方法

12.2.1　使用在线编辑器

访问 Markmap 官方在线编辑器：https://markmap.js.org/repl

12.2.2　使用 markmap-cli

1. 确保系统安装 Node.js（版本　10）。
2. 全局安装 Markmap[2]：

[1] https://markmap.js.org/

[2] Node 14 兼容的版本为 markmap-cli 0.14，而 markmap-cli 0.15 到 0.18 需要 Node 18

```
npm install -g markmap-cli@0.14
```

3. 运行命令将 Markdown 文件转换为思维导图：

```
markmap sample.md
```

   - 输出为 `sample.html`，自动在浏览器打开。
   - 可选参数：
     - `-o output.html`：指定输出文件名
     - `--enable-mathjax`：启用 MathJax 渲染数学公式
     - `--enable-prism`：启用 PrismJS 代码高亮
     - `--no-open`：生成后不自动打开浏览器

12.2.3 使用 VS Code 插件

1. 在 VS Code 中安装扩展：搜索 “Markmap” 并安装。
2. 创建以 `.mm.md` 结尾的 Markdown 文件（或普通 `.md` 文件）。
3. 编写 Markdown 文本，点击右上角的 “Markmap” 图标或使用快捷键（默认 `Ctrl+Alt+M`）预览思维导图。
4. 可点击 “Export” 按钮导出为 HTML。

参考资料 References

- Pandoc User's Guide
- Markdown Guide
- Markdown 指南中文版
- A writer's guide to Pandoc's Markdown
- Introducing Markdown and Pandoc