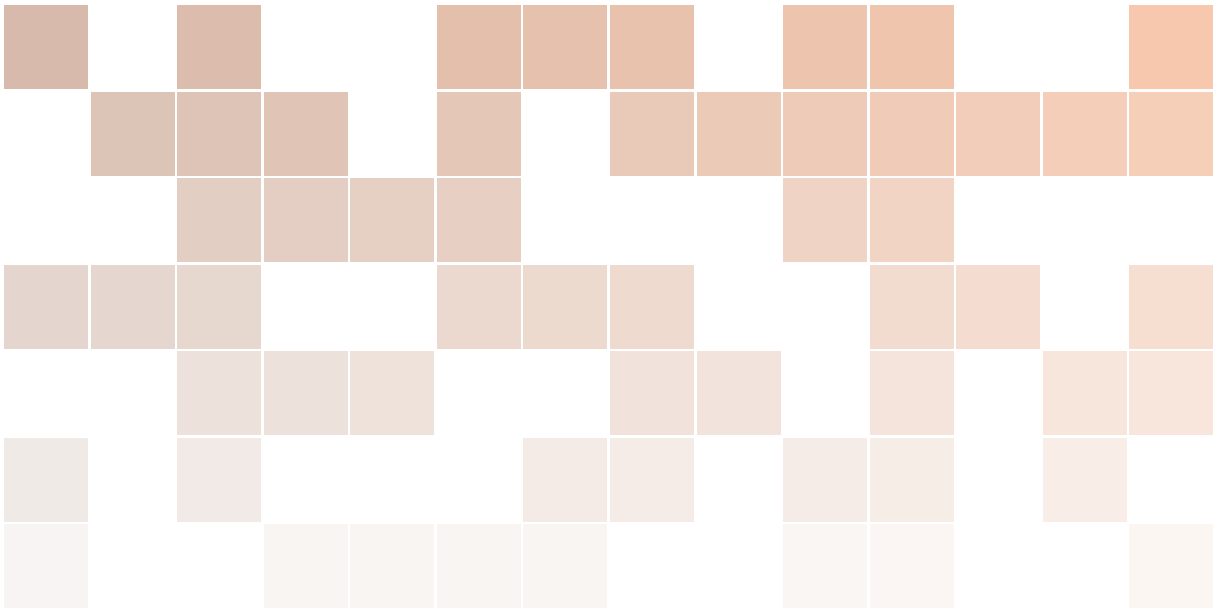
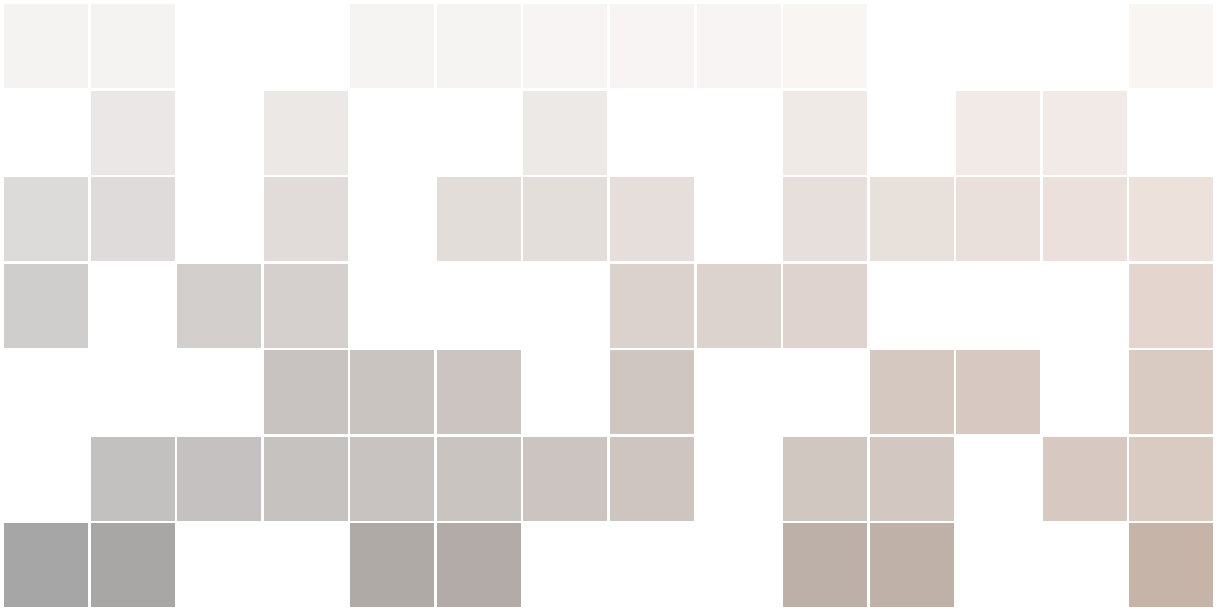


用 Markdown+Pandoc+XeLaTeX 写作

An He



PUBLISHED BY LaTeX
HTTPS://GITHUB.COM/ANNPROG/PANDOC-TEMPLATE

最后编译日期, *2019* 年 *3* 月 *30* 日 *14:47:00*

前言

LaTeX 可以排版格式精美的书籍，但是学习成本较高，使用不便；Markdown 是一种简单易学的标记语言。如果能结合两者的优点，使用 Markdown 来写作，然后通过程序转换为 latex 源码，再编译为 PDF，那将是一件美妙的事情。幸运的是，已经有工具很好的实现了支持这一功能，那就是 Pandoc。

Pandoc 是由 John MacFarlane 教授开发的标记语言转换工具，实现了数十种标记语言之间的转换。Pandoc 还扩展了 Markdown 语法，比如标题表格等的 ID 属性，脚注等，并且可以直接嵌入 LaTeX 代码，这样在 Markdown 中就可以实现输入数学公式，交叉引用等功能。Pandoc 还支持自定义转换模板，通过命令 `pandoc -D latex` 可以输出默认的 LaTeX 模板，以此模板为基础，可以定制自己的模板。

本工具定义了一种 Markdown 源码组织规范，提供了一个转换脚本，用来更方便的使用 Pandoc 将 Markdown 转换为 PDF。另外还定义了一套 LaTeX 书籍模板，用来生成中文书籍。用户也可以在自己的工作目录修改此模板，并通过修改配置来引用自己的模板。

目录

表格

插图

第一章　使用说明

1.1　安装步骤

首先克隆代码库，将 pandoc-template 目录加入环境变量，并建立工作目录。见1.1。

代码 1.1: 初始化工作环境

```
# git clone https://github.com/annProg/pandoc-template
# mkdir workdir
# cd workdir
# panbook init # 初始化工作环境
```

目录结构如代码1.2所示。

代码 1.2: 目录规范

```
.
    book                                    # 书籍模板
        book-template.epub
    panbook                                 # 转换脚本
    build                                   # 电子书构建目录
    config.default                          # pandoc默认转换配置
    html5                                   # html5电子书模板
    README.md
    resume                                  # 简历模板
    src                                     # Markdown源码目录
        images                              # 源码涉及插图目录
        metadata.yaml                       # 书籍元数据文件
    zh-ctex                                 # ctexbook模板目录
        Pictures                            # 模板引用的图片资源
```

1.2 使用规范

1.2.1 源码命名规范

脚本中使用`ls src/*.md`列出所有的 Markdown 源码，要保证顺序正确，才能生成正确的 LaTeX 源码。因此，要求 Markdown 源码文件命名能够被 ls 以正确的顺序列出。其中，`frontmatter.md`和`backmatter.md`用于前言和后记，文件名固定不可更改。例如，有少于十个的 Markdown 文件，可以像代码1.3这样组织源码：

代码 1.3: 源码命名规范

```
$ tree src/
src/
    0-title.md
    1-intro.md
    2-pandoc-markdown.md
    3-template.md
    frontmatter.md
    backmatter.md
    metadata.yaml
    images
```

如果 Markdown 文件数多于 10 个，则需要在前缀为个位数的前面补 0，与最大前缀数字位数保持一致，例如最后一个 Markdown 文件为`99-markdown.md`，那么个位数应形如`01-first.md`。

1.2.2 编码规范

Markdown 源码文件需要使用 UTF-8 编码。以 Notepad++ 为例，依次选择格式，以 **UTF-8 无 BOM** 格式编码即可正确设置编码。

1.2.3 注意事项

Pandoc 扩展的 Markdown 语法要求在标题前留出一个空行，因此按章节拆分的多个 Markdown 文件，开头需要空一行，否则 pandoc 不能正确识别标题。

1.3 书籍元数据

在`src/metadata.yaml`中使用 Yaml 语言[1] 定义书籍的数据及可用的模板变量，如代码1.4所示。

查看模板文件，可以获取所有变量（形如`$var$`）。也可以通过修改模板来添加自定义的变量。

[1] http://www.ruanyifeng.com/blog/2016/07/yaml.html

代码 1.4: code:template-var

```
---
title: 用Markdown+Pandoc+XeLaTeX写作
author:          # 作者（数组）
  - An He
date: \today     # 日期
copyright: true  # 是否生成版权页
lof: true        # 是否生成插图列表页
lot: true        # 是否生成表格列表页
graphics: true   # 是否使用graphicx
homepage: https://github.com/annProg/pandoc-template
header-includes:
  - \usepackage{cleveref}
...
```

1.4 前言后记

在frontmatter.md中添加前言，backmatter.md中添加后记。对于epub电子书，可以给标题添加epub type属性，见代码1.5。

代码 1.5: code:epub-type-attr

```
# My chapter {epub:type=prologue}
```

支持如下属性如表1.1：

表 1.1: epub:type of first section epub:type of body

Attr	Type
prologue	frontmatter
abstract	frontmatter
acknowledgments	frontmatter
copyright-page	frontmatter
dedication	frontmatter
foreword	frontmatter
halftitle	frontmatter
introduction	frontmatter
preface	frontmatter
seriespage	frontmatter
titlepage	frontmatter
afterword	backmatter

Attr	Type
appendix	backmatter
colophon	backmatter
conclusion	backmatter
epigraph	backmatter

1.5 转换命令

pandoc-template 目录加入环境变量后可以直接调用`panbook`，如代码1.6：

代码 1.6: 转换命令

```
panbook init  # 初始化工作环境
panbook pdf   # 生成pdf电子书
panbook html  # 生成html电子书
panbook pdf d # 调试模式，只使用一个代码高亮风格， html电子书也支持调试模式
```

第二章　Pandoc Markdown 语法简介

Pandoc 的目标与原始 Markdown 的最初目标有着方向性的不同。在 Markdown 原本的设计中，HTML 是其主要输出对象；然而 Pandoc 则是针对多种输出格式而设计。因此，虽然 Pandoc 同样也允许直接嵌入 HTML 标签，但并不鼓励这样的作法，取而代之的是 Pandoc 提供了许多非 HTML 的方式，来让使用者输入像是定义列表、表格、数学公式以及脚注等诸如此类的重要文件元素。

Pandoc Markdown 语法介绍可以在 Pandoc 主页[1] 找到。以下翻译大部分部分摘自 tzengyuxiao 的翻译[2]，在此向译者表示感谢。

2.1　段落

一个段落指的是一行以上的文字，跟在一行以上的空白行之后。换行字元会被当作是空白处理，因此你可以依自己喜好排列段落文字。如果你需要强制换行，在行尾放上两个以上的空白字元即可。

Extension: escaped_line_breaks

一个反斜线后跟着一个换行字元，同样也有强制换行的效果。这也是在表格单元格中添加换行的唯一形式。

2.2　标题

有两种不同形式的标题语法，Setext 以及 Atx。Setext 风格的标题是由一行文字底下接着一行 = 符号（用于一阶标题）或-符号（用于二阶标题）所构成；Atx 风格的标题是由一到六个 # 符号以及一行文字所组成，你可以在文字后面加上任意数量的 # 符号。由行首起算的 # 符号数量决定了标题的阶层，如代码2.1所示。

Extension: blank_before_header

原始 markdown 语法在标题之前并不需要预留空白行。Pandoc 则需要（除非标题位于文件最开始的地方）。这是因为以 # 符号开头的情况在一般文字段落中相当常见，这会导致非预期的标题。例如：

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

[1] http://www.pandoc.org/MANUAL.html#pandocs-markdown

[2] http://pages.tzengyuxio.me/pandoc/

代码 2.1: 测试

```
Setext A level-one header
==================

Setext A level-two header
------------------

# Atx level-one

## Atx level-two

### Atx  level-three
```

这也是前一章所述注意事项1.2.3的原因。

2.2.1 标题标识符

Extension: header_attributes

在标题文字所在行的行尾，可以使用以下语法为标题加上属性：

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

虽然这个语法也包含加入类别 (class) 以及键 / 值形式的属性 (attribute)，但目前只有标识符 (identifier/ID) 在输出时有实际作用（且只在部分格式的输出，包括：HTML, LaTeX, ConTeXt, Textile, AsciiDoc）。举例来说，下面是将标题加上 foo 标识符的几种方法：

```
# My header {#foo}

## My header ##    {#foo}

My other header   {#foo}
---------------
```

（此语法与 PHP Markdown Extra 相容。）

具有 unnumbered 类别的标题将不会被编号，即使–number-sections 的选项是开启的。单一连字符号 (-) 等同于.unnumbered，且更适用于非英文文件中。因此，

```
# My header {-}
```

与下面这行是等价的

```
# My header {.unnumbered}
```

2.3 引用

Markdown 使用 email 的习惯来建立引用区块。一个引用区块可以由一或多个段落或其他的区块元素（如列表或标题）组成，并且其行首均是由一个 > 符号加上一个空白作为开头。（> 符号不一定要位在该行最左边，但也不能缩进超过三个空白）。

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

效果如下：

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

有一个「偷懒」的形式：你只需要在引用区块的第一行行首输入 > 即可，后面的行首可以省略符号：

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

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

由于区块引用可包含其他区块元素，而区块引用本身也是区块元素，所以，引用是可以嵌套入其他引用的。

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

Extension: blank_before_blockquote

原始 markdown 语法在区块引用之前并不需要预留空白行。Pandoc 则需要（除非区块引用位于文件最开始的地方）。这是因为以 > 符号开头的情况在一般文字段落中相当常见（也许由于断行所致），这会导致非预期的格式。因此，除非是指定为 markdown_strict 格式，不然以下的语法在 pandoc 中将不会产生出嵌套区块引用：

```
> This is a block quote.
>> Nested.
```

2.4 代码

2.4.1 缩进式代码块

由四个空格或一个 tab 缩进的文本取做代码块，区块中的特殊字符、空格和换行都会被保留，而缩进的空格和 tab 会在输出中移除，但在代码块中的空行不必缩进。

```
#!/bin/bash

echo "Hello Markdown"
echo "Hello LaTeX"
```

2.4.2 围栏式代码块

Extension: fenced_code_blocks

除了标准的缩进式代码块之外，Pandoc 还支持围栏式代码块，代码块以三个或三个以上的 ~ 符号行开始，以等于或多于开始行 ~ 个数符号行结束，若是代码块中含有 ~，只需使开始行和结束行中的 ~ 符号个数多于代码块中的即可

```
~~~~~
~~~~
code here
~~~~
~~~~~~
```

Extension: backtick_code_blocks

与`fenced_code_blocks`相同，只不过使用反引号 ‘ 替换波浪线 ~ 而已

Extension: fenced_code_attributes

代码 2.2: 围栏式代码块

```
100 ~~~~ {#code:mycode .haskell .numberLines startFrom="100" caption="围栏式代码块"}
101 qsort []     = []
102 qsort (x:xs) = qsort (filter (< x) xs) ++ [x] ++
103                qsort (filter (>= x) xs)
104 ~~~~~~
```

这里的 mycode 为 ID，haskell 与 numberLines 是类别，而 startsFrom 则是值为 100 的属性。numberLines 和 startFrom 配合使用可以显示代码行号，如果没有指定 startFrom，则默认从 1 开始。caption 指定代码块标题，如果没有设置 caption，则默认使用 ID 作为 caption。有些输出格式可以利用这些信

息来作语法高亮。目前使用到这些信息的输出格式仅有 HTML 与 LaTeX。如果指定的输出格式及语言类别有语法高亮支持，那么上面那段代码区块将会以高亮并带有行号的方式呈现。

仅指定高亮语言时，可以简写为以下形式：

```
~~~haskell
qsort [] = []
~~~
```

2.5 行区块

Extension: line_blocks

行区块是一连串以竖线 (|) 加上一个空格所构成的连续行。行与行间的区隔在输出时将会以原样保留，行首的空白字元数目也一样会被保留；反之，这些行将会以 markdown 的格式处理。这个语法在输入诗句或地址时很有帮助。

```
| 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
```

如果有需要的话，书写时也可以将完整一行拆成多行，但后续行必须以空白作为开始。下面范例的前两行在输出时会被视为一整行：

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

效果：

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

这是从 reStructuredText 借来的语法。

2.6 列表

2.6.1 无序列表

无序列表是以项目符号作列举的列表。每条项目都以项目符号 (*, + 或-) 作开头。下面是个简单的例子：

```
* one
* two
* three
```

显示如下

- one
- two
- three

这会产生一个「紧凑」列表。如果你想要一个「宽松」列表，也就是说以段落格式处理每个项目内的文字内容，那么只要在每个项目间加上空白行即可：

```
* one

* two

* three
```

项目符号不能直接从行首最左边处输入，而必须以一至三个空白字元作缩进。项目符号后必须跟着一个空白字元。

列表项目中的接续行，若与该项目的第一行文字对齐（在项目符号之后），看上去会较为美观：

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

但 markdown 也允许以下「偷懒」的格式：

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

四个空白规则

一个列表项目可以包含多个段落以及其他区块等级的内容。然而，后续的段落必须接在空白行之后，并且以四个空白或一个 tab 作缩进。因此，如果项目里第一个段落与后面段落对齐的话（也就是项目符号前置入两个空白），看上去会比较整齐美观：

```
  * First paragraph.

    Continued.

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

        { code }
```

列表项目也可以包含其他列表。在这情况下前置的空白行是可有可无的。嵌套列表必须以四个空白或一个 tab 作缩进：

```
* fruits
    + apples
        - macintosh
        - red delicious
    + pears
    + peaches
* vegetables
    + brocolli
    + chard
```

上一节提到，markdown 允许你以「偷懒」的方式书写，项目的接续行可以不和第一行对齐。不过，如果一个列表项目中包含了多个段落或是其他区块元素，那么每个元素的第一行都必须缩进对齐。

```
+ A lazy, lazy, list
item.

+ Another one; this looks
bad but is legal.

    Second paragraph of second
list item.
```

注意：尽管针对接续段落的「四个空白规则」是出自于官方的 markdown syntax guide，但是作为对应参考用的 Markdown.pl 实作版本中并未遵循此一规则。所以当输入时若接续段落的缩进少于四个空白时，pandoc 所输出的结果会与 Markdown.pl 的输出有所出入。

在 markdown syntax guide 中并未明确表示「四个空白规则」是否一体适用于所有位于列表项目里的区块元素上；规范文件中只提及了段落与代码区块。但文件暗示了此规则适用于所有区块等级的内容（包含嵌套列表），并且 pandoc 以此方向进行解读与实作。

2.6.2 有序列表

有序列表与无序列表相类似，唯一的差别在于列表项目是以列举编号作开头，而不是项目符号。

在原始 markdown 中，列举编号是阿拉伯数字后面接着一个句点与空白。数字本身代表的数值会被忽略，因此下面两个列表并无差别：

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

上下两个列表的输出是相同的。

```
5.  one
7.  two
1.  three
```

Extension: fancy_lists

与原始 markdown 不同的是，Pandoc 除了使用阿拉伯数字作为有序列表的编号外，也可以使用大写或小写的英文字母，以及罗马数字。列表标记可以用括号包住，也可以单独一个右括号，抑或是句号。如果列表标记是大写字母接着一个句号，句号后请使用至少两个空白字元。

Extension: startnum

除了列表标记外，Pandoc 也能判读列表的起始编号，这两项资讯都会保留于输出格式中。举例来说，下面的输入可以产生一个从编号 9 开始，以单括号为编号标记的列表，底下还跟着一个小写罗马数字的子列表：

```
 9)  Ninth
10)  Tenth
11)  Eleventh
       i. subone
      ii. subtwo
     iii. subthree
```

当遇到不同形式的列表标记时，Pandoc 会重新开始一个新的列表。所以，以下的输入会产生三份列表：

```
(2) Two
(5) Three
1.  Four
*   Five
```

如果需要预设的有序列表标记符号，可以使用 `#.`：

```
#.  one
#.  two
#.  three
```

2.6.3 定义列表

Extension: definition_lists

Pandoc 支援定义列表，其语法的灵感来自于 PHP Markdown Extra 以及 reStructuredText：

```
Term 1

:   Definition 1

Term 2 with *inline markup*

:   Definition 2

        { some code, part of Definition 2 }

    Third paragraph of definition 2.
```

每个专有名词 (term) 都必须单独存在于一行，后面可以接着一个空白行，也可以省略，但一定要接上一或多笔定义内容。一笔定义需由一个冒号或波浪线作开头，可以接上一或两个空白作为缩进。定义本身的内容主体（包括接在冒号或波浪线后的第一行）应该以四个空白缩进。一个专有名词可以有多个定义，而每个定义可以包含一或多个区块元素（段落、代码区块、列表等），每个区块元素都要缩进四个空白或一个 tab。

如果你在定义内容后面留下空白行（如同上面的范例），那么该段定义会被当作段落处理。在某些输出格式中，这意谓著成对的专有名词与定义内容间会有较大的空白间距。在定义与定义之间，以及定义与下个专有名词间不要留空白行，即可产生一个比较紧凑的定义列表：

```
Term 1
  ~ Definition 1
Term 2
  ~ Definition 2a
  ~ Definition 2b
```

2.6.4 编号范例列表

Extension: example_lists

这个特别的列表标记 @ 可以用来产生连续编号的范例列表。列表中第一个以 @ 标记的项目会被编号为'1'，接着编号为'2'，依此类推，直到文件结束。范例项目的编号不会局限于单一列表中，而是文件中所有以 @ 为标记的项目均会次序递增其编号，直到最后一个。举例如下：

```
(@)  My first example will be numbered (1).
(@)  My second example will be numbered (2).

Explanation of examples.
```

```
(@)  My third example will be numbered (3).
```

编号范例可以加上标签，并且在文件的其他地方作参照：

```
(@good)  This is a good example.

As (@good) illustrates, ...
```

标签可以是由任何英文字母、底线或是连字符号所组成的字串。

2.6.5 紧凑与宽松列表

在与列表相关的「边界处理」上，Pandoc 与 Markdown.pl 有着不同的处理结果。考虑如下代码：

```
+   First
+   Second:
    -   Fee
    -   Fie
    -   Foe

+   Third
```

Pandoc 会将以上列表转换为「紧凑列表」（在 “First”, “Second” 或 “Third” 之中没有 <p> 标签），而 markdown 则会在 “Second” 与 “Third”（但不包含 “First”）里面置入 <p> 标签，这是因为 “Third” 之前的空白行而造成的结果。Pandoc 依循着一个简单规则：如果文字后面跟着空白行，那么就会被视为段落。既然 “Second” 后面是跟着一个列表，而非空白行，那么就不会被视为段落了。至于子列表的后面是不是跟着空白行，那就无关紧要了。（注意：即使是设定为 markdown_strict 格式，Pandoc 仍是依以上方式处理列表项目是否为段落的判定。这个处理方式与 markdown 官方语法规范里的描述一致，然而却与 Markdown.pl 的处理不同。）

2.6.6 结束一个列表

如果你在列表之后放入一个缩排的代码区块，会有什么结果？

```
-   item one
-   item two

    { my code block }
```

问题大了！这边 pandoc（其他的 markdown 实作也是如此）会将 { my code block } 视为 item two 这个列表项目的第二个段落来处理，而不会将其视为一个代码区块。

要在 item two 之后「切断」列表，你可以插入一些没有缩排、输出时也不可见的内容，例如 HTML 的注解：

```
-   item one
-   item two

```

```
<!-- end of list -->

    { my code block }
```

当你想要两个各自独立的列表，而非一个大且连续的列表时，也可以运用同样的技巧：

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

<!-- -->

1.  uno
2.  dos
3.  tres
```

2.7 分隔线

一行中若包含三个以上的 *, -或 _ 符号（中间可以以空白字元分隔），则会产生一条分隔线：

```
*  *  *  *

---------------
```

2.8 表格

有四种表格的形式可以使用。前三种适用于等宽字型的编辑环境，例如 Courier。第四种则不需要直行的对齐，因此可以在比例字型的环境下使用。

2.8.1 简单表格

Extension: simple_tables,table_captions

简单表格看起来如表2.1所示：

表 2.1: Demonstration of simple table syntax.

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

代码为：

```
简单表格看起来如表\ref{table:simpletable}所示：

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

Table:  Demonstration of simple table syntax.\label{table:simpletable}
```

可以用`\label`为表格添加 label，然后在其他地方用`\ref`引用。

表头与资料列分别以一行为单位。直行的对齐则依照表头的文字和其底下虚线的相对位置来决定：

- 如果虚线与表头文字的右侧有切齐，而左侧比表头文字还长，则该直行为靠右对齐。
- 如果虚线与表头文字的左侧有切齐，而右侧比表头文字还长，则该直行为靠左对齐。
- 如果虚线的两侧都比表头文字长，则该直行为置中对齐。
- 如果虚线与表头文字的两侧都有切齐，则会套用预设的对齐方式（在大多数情况下，这将会是靠左对齐）。
- 表格底下必须接着一个空白行，或是一行虚线后再一个空白行。表格标题为可选的（上面的范例中有出现）。标题需是一个以 Table:（或单纯只有:）开头作为前缀的段落，输出时前缀的这部份会被去除掉。表格标题可以放在表格之前或之后。

表头也可以省略，在省略表头的情况下，表格下方必须加上一行虚线以清楚标明表格的范围。例如：

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

代码：

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

当省略表头时，直行的对齐会以表格内容的第一行资料列决定。所以，以上面的表格为例，各直行的对齐依序会是靠右、靠左、置中以及靠右对齐。

2.8.2 多行表格

Extension: multiline_tables,table_captions

多行表格允许表头与表格资料格的文字能以多行呈现（但不支援横跨多栏或纵跨多列的资料格）。以下为范例：

表 2.3: 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.

代码：

```
-------------------------------------------------------------
 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.
```

看起来很像简单表格，但两者间有以下差别：

- 在表头文字之前，必须以一列虚线作为开头（除非有省略表头）。
- 必须以一列虚线作为表格结尾，之后接一个空白行。
- 资料列与资料列之间以空白行隔开。
- 在多行表格中，表格分析器会计算各直行的栏宽，并在输出时尽可能维持各直行在原始文件中的相对比例。因此，要是你觉得某些栏位在输出时不够宽，你可以在 markdown 的原始档中加宽一点。

和简单表格一样，表头在多行表格中也是可以省略的：

表 2.4: Here's a multiline table without headers.

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.

代码：

```
----------- ------- --------------- -------------------------
   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 headers.
```

多行表格中可以单只包含一个资料列，但该资料列之后必须接着一个空白行（然后才是标示表格结尾的一行虚线）。如果没有此空白行，此表格将会被解读成简单表格。

2.8.3 格框表格

Extension: grid_tables,table_captions

格框表格看起来像这样：

表 2.5: Sample grid table.

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

代码：

```
: Sample grid table.
```

```
+---------------+---------------+--------------------+
| Fruit         | Price         | Advantages         |
+===============+===============+====================+
| Bananas       | $1.34         | - built-in wrapper |
|               |               | - bright color     |
+---------------+---------------+--------------------+
| Oranges       | $2.10         | - cures scurvy     |
|               |               | - tasty            |
+---------------+---------------+--------------------+
```

以 = 串成的一行区分了表头与表格本体，这在没有表头的表格中也是可以省略的。在格框表格中的资料格可以包含任意的区块元素（复数段落、代码区块、清单等等）。不支援对齐，也不支援横跨多栏或纵跨多列的资料格。格框表格可以在 Emacs table mode 下轻松建立。

2.8.4 管线表格

Extension: pipe_tables,table_captions

管线表格看起来像这样：

表 2.6: Demonstration of simple table syntax.

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

代码：

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

  : Demonstration of simple table syntax.
```

这个语法与 PHP markdown extra 中的表格语法相同。开始与结尾的管线字元是可选的，但各直行间则必须以管线区隔。上面范例中的冒号表明了对齐方式。表头可以省略，但表头下的水平虚线必须保留，因为虚线上定义了资料栏的对齐方式。

因为管线界定了各栏之间的边界，表格的原始码并不需要像上面例子中各栏之间保持直行对齐。所以，底下一样是个完全合法（虽然丑陋）的管线表格：

```
fruit| price
```

```
-----|-----:
apple|2.05
pear|1.37
orange|3.09
```

管线表格的资料格不能包含如段落、清单之类的区块元素，也不能包含多行文字。

注意：Pandoc 也可以看得懂以下形式的管线表格，这是由 Emacs 的 orgtbl-mod 所绘制：

```
| One | Two   |
|-----+-------|
| my  | table |
| is  | nice  |
```

效果：

One	Two
my	table
is	nice

主要的差别在于以 + 取代了部分的 |。其他的 orgtbl 功能并未支援。如果要指定非预设的直行对齐形式，你仍然需要在上面的表格中自行加入冒号。

2.9 文件标题区块

（译注：本节中提到的「标题」均指 Title，而非 Headers）

Extension: pandoc_title_block

如果档案以文件标题（Title）区块开头

```
% title
% author(s) (separated by semicolons)
% date
```

这部份将不会作为一般文字处理，而会以书目资讯的方式解析。（这可用在像是单一 LaTeX 或是 HTML 输出文件的书名上。）这个区块仅能包含标题，或是标题与作者，或是标题、作者与日期。如果你只想包含作者却不想包含标题，或是只有标题与日期而没有作者，你得利用空白行：

```
%
% Author

% My title
%
% June 15, 2006
```

标题可以包含多行文字，但接续行必须以空白字元开头，像是：

```
% My title
  on multiple lines
```

如果文件有多个作者，作者也可以分列在不同行并以空白字元作开头，或是以分号间隔，或是两者并行。所以，下列各种写法得到的结果都是相同的：

```
% Author One
  Author Two

% Author One; Author Two

% Author One;
  Author Two
```

日期就只能写在一行之内。

所有这三个 metadata 栏位都可以包含标准的行内格式（斜体、连结、脚注等等）。

文件标题区块一定会被分析处理，但只有在`--standaline( -s)`选项被设定时才会影响输出内容。在输出 HTML 时，文件标题会出现的地方有两个：一个是在文件的`<head>`区块里—这会显示在浏览器的视窗标题上—另外一个是文件的`<body>`区块最前面。位于`<head>`里的文件标题可以选择性地加上前缀文字（透过`--title-prefix`或`-T`选项）。而在`<body>`里的文件标题会以 H1 元素呈现，并附带 “title” 类别 (class)，这样就能藉由 CSS 来隐藏显示或重新定义格式。如果以-T 选项指定了标题前缀文字，却没有设定文件标题区块里的标题，那么前缀文字本身就会被当作是 HTML 的文件标题。

而 man page 的输出器会分析文件标题区块的标题行，以解出标题、man page section number，以及其他页眉 (header) 页脚 (footer) 所需要的资讯。一般会假设标题行的第一个单字为标题，标题后也许会紧接着一个以括号包住的单一数字，代表 section number（标题与括号之间没有空白）。在此之后的其他文字则为页脚与页眉文字。页脚与页眉文字之间是以单独的一个管线符号 (|) 作为区隔。所以，

```
% PANDOC(1)
```

将会产生一份标题为 PANDOC 且 section 为 1 的 man page。

```
% PANDOC(1) Pandoc User Manuals
```

产生的 man page 会再加上 “Pandoc User Manuals” 在页脚处。

```
% PANDOC(1) Pandoc User Manuals | Version 4.0
```

产生的 man page 会再加上 “Version 4.0” 在页眉处。

2.10 字符转义

Extension: all_symbols_escapable

除了在代码区块或行内代码之外，任何标点符号或空白字元前面只要加上一个反斜线，都能使其保留字面原义，而不会进行格式的转义解读。因此，举例来说，下面的写法

```
*\*hello\**
```

输出后会得到

```
<em>*hello*</em>
```

而不是

```
<strong>hello</strong>
```

这条规则比原始的 markdown 规则来得好记许多，原始规则中，只有以下字元才支持反斜线跳脱，不作进一步转义：

```
\`*_{}[]()>#+-.!
```

（然而，如果使用了 markdown_strict 格式，那么就会采用原始的 markdown 规则）

一个反斜线之后的空白字元会被解释为不断行的空白 (nonbreaking space)。这在 TeX 的输出中会显示为~，而在 HTML 与 XML 则是显示为\ 或\ 。

一个反斜线之后的换行字元（例如反斜线符号出现在一行的最尾端）则会被解释为强制换行。这在 TeX 的输出中会显示为\\，而在 HTML 里则是
。相对于原始 markdown 是以在行尾加上两个空白字元这种「看不见」的方式进行强制换行，反斜线接换行字元会是比较好的替代方案。

转义字符在代码上下文中不起任何作用。

2.11 智能标点

如果指定了--smart选项,pandoc 将会输出正式印刷用的标点符号,像是将straight quotes转换为curly quotes [3]、---转为破折号 (em-dashes)，--转为连接号 (en-dashes)，以及将 ...转为省略号。不断行空格 (Nonbreaking spaces) 将会插入某些缩写词之后，例如 “Mr.”。

注意：如果你的 LaTeX template 使用了 csquotes 套件，pandoc 会自动侦测并且使用\enquote{...}在引言文字上。

2.12 行内格式

强调要强调某些文字，只要以*或_符号前后包住即可，像这样：

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

重复两个*或_符号以产生更强烈的强调：

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

[3]译注：straight quotes 指的是左右两侧都长得一样的引号，例如我们直接在键盘上打出来的单引号或双引号；curly quotes 则是左右两侧不同，有从两侧向内包夹视觉效果的引号。

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

一个前后以空白字元包住，或是前面加上反斜线的*或_符号，都不会转换为强调格式：

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

Extension: intraword_underscores

因为 _ 字元有时会使用在单字或是 ID 之中，所以 pandoc 不会把被字母包住的 _ 解读为强调标记。如果有需要特别强调单字中的一部分，就用 *：

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

2.12.1 删除线

Extension: strikeout

要将一段文字加上水平线作为删除效果，将该段文字前后以~~包住即可。例如，

```
This ~~is deleted text.~~
```

2.12.2 上标与下标

Extension: superscript,subscript

要输入上标可以用^字元将要上标的文字包起来；要输入下标可以用~字元将要下标的文字包起来。直接看范例，

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

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

如果要上标或下标的文字中包含了空白，那么这个空白字元之前必须加上反斜线。（这是为了避免一般使用下的~和^在非预期的情况下产生出意外的上标或下标。）所以，如果你想要让字母 P 后面跟着下标文字'a cat'，那么就要输入`P~a\ cat~`，而不是`P~a cat~`。

2.12.3 字面文字

要让一小段文字直接以其字面形式呈现，可以用反引号将其包住：

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

如果字面文字中也包含了反引号，那就使用双重反引号包住：

```
Here is a literal backtick `` ` ``.
```

（在起始反引号后的空白以及结束反引号前的空白都会被忽略。）

一般性的规则如下，字面文字区段是以连续的反引号字元作为开始（反引号后的空白字元为可选），一直到同样数目的反引号字元出现才结束（反引号前的空白字元也为可选）。

要注意的是，转义字符（以及其他 markdown 结构）在字面文字的上下文中是没有效果的：

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

Extension: inline_code_attributes

与围栏代码区块一样，字面文字也可以附加属性：

```
`<$>`{.haskell}
```

2.13 数学

Extension: tex_math_dollars

所有介于两个`$`字元之间的内容将会被视为 TeX 数学公式处理。开头的`$`右侧必须立刻接上任意文字，而结尾`$`的左侧同样也必须紧挨着文字。这样一来，`$20,000 and $30,000`就不会被当作数学公式处理了。如果基于某些原因，有必须使用`$`符号将其他文字括住的需求时，那么可以在`$`前使用反斜线跳脱字元，这样`$`就不会被当作数学公式的分隔符。

TeX 数学公式会在所有输出格式中印出。至于会以什么方式演算编排 (render) 则取决于输出的格式：

Markdown, LaTeX, Org-Mode, ConTeXt

公式会以字面文字呈现在两个 $ 符号之间。

reStructuredText

公式会使用此处所描述的:math: 这个 “interpreted text role” 来进行演算编排。

AsciiDoc

公式会以 latexmath:[...] 演算编排。

Texinfo

公式会在 @math 指令中演算编排。

groff man

公式会以去掉`$`后的字面文字演算编排。

MediaWiki

公式会在 <math> 标签中演算编排。

Textile

公式会在`<span class="math">`标签中演算编排。

RTF, OpenDocument, ODT

如果可以的话，公式会以 unicode 字元演算编排，不然就直接使用字面字元。

Docbook

如果使用了`--mathml`旗标，公式就会在 inlineequation 或 informalequation 标签中使用 mathml 演算编排。否则就会尽可能使用 unicode 字元演算编排。

Docx

公式会以 OMML 数学标记的方式演算编排。

FictionBook2

如果有使用`--webtex`选项，公式会以 Google Charts 或其他相容的网路服务演算编排为图片，并下载嵌入于电子书中。否则就会以字面文字显示。

HTML, Slidy, DZSlides, S5, EPUB

公式会依照以下命令列选项的设置，以不同的方法演算编排为 HTML 代码。

预设方式是将 TeX 数学公式尽可能地以 unicode 字元演算编排，如同 RTF、DocBook 以及 OpenDocument 的输出。公式会被放在附有属性`class="math"`的 span 标签内，所以可以在需要时给予不同的样式，使其突出于周遭的文字内容。

如果使用了`--latexmathml`选项，TeX 数学公式会被显示于`$`或`$$`字元中，并放在附带 LaTeX 类别的`<span>`标签里。这段内容会用 LaTeXMathML script 演算编排为数学公式。（这个方法无法适用于所有浏览器，但在 Firefox 中是有效的。在不支援 LaTeXMathML 的浏览器中，TeX 数学公式会单纯的以两个 $ 字元间的字面文字呈现。）

如果使用了`--jsmath`选项，TeX 数学公式会放在`<span>`标签（用于行内数学公式）或`<div>`标签（用于区块数学公式）中，并附带类别属性 math。这段内容会使用 jsMath script 来演算编排。

如果使用了`--mimetex`选项，mimeTeX CGI script 会被呼叫来产生每个 TeX 数学公式的图片。这适用于所有浏览器。`--mimetex`选项有一个可选的 URL 参数。如果没有指定 URL，它会假设 mimeTeX CGI script 的位置在`/cgi-bin/mimetex.cig`。

如果使用了`--gladtex`选项，TeX 数学公式在 HTML 的输出中会被`<eq>`标签包住。产生的 htex 档案之后可以透过 gladTeX 处理，这会针对每个数学公式生成图片，并于最后生成一个包含这些图片连结的 html 档案。所以，整个处理流程如下：

```
pandoc -s --gladtex myfile.txt -o myfile.htex
gladtex -d myfile-images myfile.htex
# produces myfile.html and images in myfile-images
```

如果使用了--webtex选项，TeX 数学公式会被转换为<img>标签并连结到一个用以转换公式为图片的外部 script。公式将会编码为 URL 可接受格式并且与指定的 URL 参数串接。如果没有指定 URL，那么将会使用 Google Chart API (http://chart.apis.google.com/chart?cht=tx&chl=)。

如果使用了--mathjax选项，TeX 数学公式将会被包在\(...\)（用于行内数学公式）或\[...\]（用于区块数学公式）之间显示，并且放在附带类别 math 的标签之中。这段内容会使用 MathJax script 演算编排为页面上的数学公式。

2.14 Raw HTML

Extension: raw_html

Markdown 允许你在文件中的任何地方插入原始 HTML（或 DocBook）指令（除了在字面文字上下文处，此时的<, >和 & 都会按其字面意义显示）。（技术上而言这不算扩充功能，因为原始 markdown 本身就有提供此功能，但做成扩充形式便可以在有特殊需要的时候关闭此功能。）

输出 HTML, S5, Slidy, Slideous, DZSlides, EPUB, Markdown 以及 Textile 等格式时，原始 HTML 代码会不作修改地保留至输出档案中；而其他格式的输出内容则会将原始 HTML 代码去除掉。

Extension: markdown_in_html_blocks

原始 markdown 允许你插入 HTML「区块」：所谓的 HTML 区块是指，上下各由一个空白行所隔开，开始与结尾均由所在行最左侧开始的一连串对称均衡的 HTML 标签。在这个区块中，任何内容都会当作是 HTML 来分析，而不再视为 markdown；所以（举例来说），* 符号就不再代表强调。

当指定格式为 markdown_strict 时，Pandoc 会以上述方式处理；但预设情况下，Pandoc 能够以 markdown 语法解读 HTML 区块标签中的内容。举例说明，Pandoc 能够将底下这段

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

转换为

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

而 Markdown.pl 则是保留该段原样。

这个规则只有一个例外：那就是介于<script>与<style>之间的文字都不会被拿来当作 markdown 解读。

这边与原始 markdown 的分歧，主要是为了让 markdown 能够更便利地混入 HTML 区块元素。比方说，一段 markdown 文字可以用 标签将其前后包住来进行样式指定，而不用担心里面的 markdown 不会被解译到。

2.15 Raw TeX

Extension: raw_tex

除了 HTML 之外，pandoc 也接受文件中嵌入原始 LaTeX, TeX 以及 ConTeXt 代码。行内 TeX 指令会被保留并不作修改地输出至 LaTeX 与 ConTeXt 格式中。所以，举例来说，你可以使用 LaTeX 来导入 BibTeX 的引用文献：

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

请注意在 LaTeX 环境下时，像是底下

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

位在 begin 与 end 标签之间的内容，都会被当作是原始 LaTeX 资料解读，而不会视为 markdown。

行内 LaTeX 在输出至 Markdown, LaTeX 及 ConTeXt 之外的格式时会被忽略掉。

2.16 LaTeX 巨集

Extension: latex_macros

当输出格式不是 LaTeX 时，pandoc 会分析 LaTeX 的\newcommand和\renewcommand定义，并套用其产生的巨集到所有 LaTeX 数学公式中。所以，举例来说，下列指令对于所有的输出格式均有作用，而非仅仅作用于 LaTeX 格式：

```
\newcommand{\tuple}[1]{\langle #1 \rangle}

$\tuple{a, b, c}$
```

在 LaTeX 的输出中，\newcommand定义会单纯不作修改地保留至输出结果。

2.17 连结

Markdown 接受以下数种指定连结的方式。

2.17.1 自动连结

如果你用角括号将一段 URL 或是 email 位址包起来，它会自动转换成连结：

http://google.com

sam@green.eggs.ham[4]

2.17.2 行内连结

一个行内连结包含了位在方括号中的连结文字，以及方括号后以圆括号包起来的 URL。（你可以选择性地在 URL 后面加入连结标题，标题文字要放在引号之中。）

```
This is an [inline link](/url), and here's [one with
a title](http://fsf.org "click here for a good time!").
```

This is an inline link[5], and here's one with a title[6].

方括号与圆括号之间不能有空白。连结文字可以包含格式（例如强调），但连结标题则否。

2.17.3 参考连结

一个明确的参考连结包含两个部分，连结本身以及连结定义，其中连结定义可以放在文件的任何地方（不论是放在连结所在处之前或之后）。

连结本身是由两组方括号所组成，第一组方括号中为连结文字，第二组为连结标签。（在两个方括号间可以有空白。）连结定义则是以方括号框住的连结标签作开头，后面跟着一个冒号一个空白，再接着一个 URL，最后可以选择性地（在一个空白之后）加入由引号或是圆括号包住的连结标题。

以下是一些范例：

```
[my label 1]: /foo/bar.html  "My title, optional"
[my label 2]: /foo
[my label 3]: http://fsf.org (The free software foundation)
[my label 4]: /bar#special  'A title in single quotes'
```

连结的 URL 也可以选择性地以角括号包住：

```
[my label 5]: <http://foo.bar.baz>
```

连结标题可以放在第二行，效果见 my label 3[7]：

```
[my label 3]: http://fsf.org
  "The free software foundation"
```

[4] mailto:sam@green.eggs.ham

[5] /url

[6] http://fsf.org

[7] http://fsf.org

需注意连结标签并不区分大小写。所以下面的例子会建立合法的连结：

```
Here is [my link][FOO]

[Foo]: /bar/baz
```

在一个隐性参考连结中，第二组方括号的内容是空的，甚至可以完全地略去：

```
See [my website][], or [my website].

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

注意：在 Markdown.pl 以及大多数其他 markdown 实作中，参考连结的定义不能存在于嵌套结构中，例如清单项目或是区块引言。Pandoc lifts this arbitrary seeming restriction。所以虽然下面的语法在几乎所有其他实作中都是错误的，但在 pandoc 中可以正确处理：

```
> My block [quote].
>
> [quote]: /foo
```

2.17.4 内部连结

要连结到同一份文件的其他章节，可使用自动产生的 ID（参见 HTML, LaTeX 与 ConTeXt 的标题识别符一节后半）。

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

或是

```
See the [Introduction].

[Introduction]: #introduction
```

内部连结目前支援的格式有 HTML（包括 HTML slide shows 与 EPUB）、LaTeX 以及 ConTeXt。

2.18 图片

在连结语法的前面加上一个! 就是图片的语法了。连结文字将会作为图片的替代文字（alt text）：

```
![la lune](Pictures/background.pdf "Voyage to the moon")

![movie reel]

[movie reel]: movie.gif
```

也可以使用 LaTeX 的`\label`为图片添加 label，这样就可以在其他地方引用了，例如：

```
效果如图\ref{fig:markdown}。

![Markdown\label{fig:markdown}](images/markdown.jpg "Markdown")
```

效果如图2.1。

2.18.1 附上说明的图片

Extension: implicit_figures

一个图片若自身单独存在一个段落中，那么将会以附上图片说明 (caption) 的图表 (figure) 形式呈现。（在 LaTeX 中，会使用图表环境；在 HTML 中，图片会被放在具有 figure 类别的 div 元素中，并会附上一个具有 caption 类别的 p 元素。）图片的替代文字同时也会用来作为图片说明。

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

如果你只是想要个一般的行内图片，那么只要让图片不是段落里唯一的元素即可。一个简单的方法是在图片后面插入一个不断行空格：

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

2.19 脚注

Extension: footnotes

Pandoc's markdown 支援脚注功能，使用如代码2.3所示的语法：

Here is a footnote reference,[8] and another.[9]

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

脚注参考用的 ID 不得包含空白、tabs 或换行字元。这些 ID 只会用来建立脚注位置与脚注文字的对应关连；在输出时，脚注将会依序递增编号。

脚注本身不需要放在文件的最后面。它们可以放在文件里的任何地方，但不能被放入区块元素（清单、区块引言、表格等）之中。

Extension: inline_notes

Pandoc 也支援了行内脚注（尽管，与一般脚注不同，行内脚注不能包含多个段落）。其语法如下：

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

[8] Here is the footnote.

[9] 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.

图 2.1: Markdown

代码 2.3: 脚注语法

```
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.
```

Here is an inline note.[10]

行内与一般脚注可以自由交错使用。

2.20 引用

Extension: citations

Pandoc 能够以数种形式自动产生引用与参考书目（使用 Andrea Rossato 的 hs-citeproc）。为了使用这项功能，你需要一个下列其中一种格式的参考书目资料库，如表2.8所示：

表 2.8: Pandoc 支持的引用形式

Format	File extension
MODS	.mods
BibLaTeX	.bib
BibTeX	.bibtex
RIS	.ris
EndNote	.enl
EndNote XML	.xml
ISI	.wos
MEDLINE	.medline

[10] Inlines notes are easier to write, since you don't have to pick an identifier and move down to type the note.

Format	File extension
Copac	.copac
JSON citeproc	.json

需注意的是副档名.bib 一般而言同时适用于 BibTeX 与 BibLaTeX 的档案，不过你可以使用.bibtex 来强制指定 BibTeX。

你需要使用命令列选项`--bibliography`来指定参考书目档案（如果有多个书目档就得反覆指定）。

预设情况下，pandoc 会在引用文献与参考书目中使用芝加哥「作者－日期」格式。要使用其他的格式，你需要用`--csl`选项来指定一个 CSL 1.0 格式的档案。关于建立与修改 CSL 格式的入门可以在 http://citationstyles.org/downloads/primer.html 这边找到。https://github.com/citation-style-language/styles 是 CSL 格式的档案库。也可以在 http://zotero.org/styles 以简单的方式浏览。

引用资讯放在方括号中，以分号区隔。每一条引用都会有个 key，由 @ 加上资料库中的引用 ID 组成，并且可以选择性地包含前缀、定位以及后缀。以下是一些范例：

```
Blah blah [see @doe99, pp. 33-35; also @smith04, ch. 1].

Blah blah [@doe99, pp. 33-35, 38-39 and *passim*].

Blah blah [@smith04; @doe99].
```

在 @ 前面的减号 (-) 将会避免作者名字在引用中出现。这可以用在已经提及作者的文章场合中：

```
Smith says blah [-@smith04].
```

你也可以在文字中直接插入引用资讯，方式如下：

```
@smith04 says blah.

@smith04 [p. 33] says blah.
```

如果引用格式档需要产生一份引用作品的清单，这份清单会被放在文件的最后面。一般而言，你需要以一个适当的标题结束你的文件：

```
last paragraph...

# References
```

如此一来参考书目就会被放在这个标题后面了。

后记

啦啦啦，感谢阅读。