Pandoc's Markdown

作者：John MacFarlane

译者：PPType

Contents

前言

本文档是 Pandoc 版 Markdown 语法的简体中文译本。Pandoc 本身是由 John MacFarlane 所开发的文档转换工具，可以在 HTML, Markdown, PDF, TeX 等格式之间进行转换。有许多喜欢纯文本编辑的人，利用 Pandoc 进行论文撰写或投影片制作。除了转换功能外，Pandoc 所定义的 Markdown 扩展语法也是这套工

具的一大亮点，在 Pandoc 的官方文档中，仅针对 Markdown 格式的扩展说明就占了一半左右的篇幅。

本文档翻译自 Pandoc - Pandoc User's Guide 中的 "Pandoc's Markdown" 一节。你可以在 GitHub 上查看本文档的源文件。

以下翻译开始。

Pandoc's Markdown

与 John Gruber 的原始 Markdown 相比，Pandoc 版本的 Markdown 在语法上进行了扩展和少许修正。本文档解释了这些语法，并指出其与标准 Markdown 的差异。除非特别提到，这些差异均可借由使用 `markdown_strict` 而非 `markdown` 的格式来消除。一项扩展可通过 `+EXTENSION` 或 `-EXTENSION` 的方式来开启或关闭。例如，`markdown_strict+footnotes` 表示加上脚注扩展的严格 Markdown，而 `markdown-footnotes-pipe_tables` 则是去掉了脚注与管线表格扩展的 pandoc Markdown。

哲学

Markdown 的设计目标是易于书写，并且更重要的是易于阅读：

> 一份 Markdown 格式的文档应该能原封不动地以纯文本形式发表，并且看上去不像是被标签或格式指令标记过。— John Gruber

Pandoc 在制订表格、脚注以及其他扩展的语法时都遵循了这项原则。

不过，pandoc 是针对多种输出格式设计的，而 Markdown 最初主要考虑的是生成 HTML，两者在这方面目标不同。因此虽然 pandoc 允许直接嵌入原始 HTML，但并不鼓励这样做，而是提供了许多非 HTML 风格的方式，让用户输入定义列表、表格、数学公式以及脚注等重要文档元素。

段落

一个段落指的是一行及以上的文本，后面紧跟着一个或多个空行。换行字符会被当作空格，你可以依据自己的喜好排列文本段落。如果你需要插入一个强制换行符，在行尾放两个或以上的空格即可。

Extension: `escaped_line_breaks`

一个反斜线后面加一个换行符，同样构成一个强制换行符。注意：在多行内容和表格的单元格中，这是创造强制换行符的唯一方法，因为单元格中，行尾的空格会被忽略。

标题

有两种标题：Setext 和 ATX。

Setext 风格标题

Setext 风格的标题上面是一行文本，下面接着一行 = 符号（用于一级标题）或 - 符号（用于二级标题）：

```
一个一级标题
==========

一个二级标题
----------
```

标题文本可以包含行内格式，例如强调（见下方行内格式 一节）。

ATX 风格标题

ATX 风格的标题由一到六个 # 符号及一行文本组成，文本后面也可以选择性地加上任意数量的 # 符号。行首的 # 符号数量决定了标题的层级：

```
## 一个二级标题

### 一个三级标题 ###
```

如同 setext 风格，ATX 风格标题的文本也可以包含行内格式：

```
# 一个一级标题和一个[链接](/url) 以及*强调*
```

Extension: `blank_before_header`

标准 Markdown 语法并不要求在标题之上留一个空行，而 pandoc Markdown 的标题上面则必须加上空行（除非标题位于文档第一行）。这是因为 `#` 符号很可能完全意外地出现在行首（也许是因为自动断行）。考虑下面的例子：

```
我喜欢好几种他们的冰淇凌口味：
#22， 比如，还有 #5。
```

标题标识符

Extension: `header_attributes`

在标题文本行的末尾，可用以下语法为标题加上属性：

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

例如，下面所有的标题都会加上 `foo` 这个 ID：

```
# 我的标题 {#foo}

## 我的标题 ## {#foo}

我的其他标题 {#foo}
---------------
```

（此语法与 PHP Markdown Extra 兼容。）

需要注意的是，虽然语法上允许使用标识符 (ID), 类 (class) 以及键/值对 (key/vaule) 形式的属性 (attribute)，但通常使用者并不会加入所有这些信息。ID，类，还有键值对这些属性，会被用在 HTML 和基于 HTML 的文件格式上，比如 EPUB 和 slidy。而在 LaTeX, ConTeXt, Textile 和 AsciiDoc 中，使用者会用 ID 来标识标签或链接。

具有 `unnumbered` 类的标题将不会被编号，即使已经指定了 `--number-sections` 。单个连字符 (`-`) 等同于 `.unnumbered`，且更适用于非英文文档。因此，

```
# 我的标题 {-}
```

等价于

```
# 我的标题 {.unnumbered}
```

Extension: `auto_identifiers`

对于没有明确指定 ID 的标题，pandoc 将依据其文本，自动指派一个独一无二的 ID。由标题文本推导 ID 的规则如下：

- 移除所有格式，链接等。
- 移除所有脚注。
- 移除所有标点符号，除了下划线、连字符与句号。
- 以连字符取代所有空格与换行字符。
- 将所有英文字母转为小写。
- 移除第一个字母前的所有内容（ID 不能以数字或标点符号开头）。
- 如果剩下的为空字符串，则使用 `section` 作为 ID。

以下是一些范例，

标题	ID
`HTML` 中的标题标识符	`html`-中的标题标识符
`* 狗狗们 *?--在 * 我的 * 家里?`	狗狗们--在-我的-家里
`[HTML], [S5], or [RTF]?`	`html-s5-or-rtf`
`3. 应用`	应用
33	`section`

大多数情况下，这些规则应该能让人直接从标题文本推导出 ID。唯一例外的是，当多个标题具有相同文本时，第一个标题的 ID 仍旧是通过以上规则得出，而第二个则是在同样 ID 后加上 `-1`，第三个加上 `-2`，以此类推。

在开启 `--toc|--table-of-contents` 的选项时，这些 ID 是用来产生目录 (Table of Contents) 所需的页面链接。此外根据这些 ID，可以很轻松地在文档中设置从一个章节到另一个章节的链接。比如本小节的链接看起来就像这样：

```
阅读章节
```

`[标题标识符](#标题标识符)`。

然而要注意的是，这种产生章节链接的方式，只在 HTML、LaTeX 和 ConTeXt 输出格式中有效。

如果指定了 `--section-divs` 选项，则每一个小节都会以 `div` 标签包住（或是 `section` 标签，如果有指定 `--html5` 选项的话），并且 ID 会被附加在用来包住小节的 `<div>`（或是 `<section>`）标签，而非附加在标题上。这使得整个小节都可以使用 JavaScript 来操作，或是采用不同的 CSS 设置。

Extension: `implicit_header_references`

Pandoc 假设每个标题都被定义了指向链接，因此要在文档中链接到以下标题：

```
# HTML 中的标题标识符
```

你可以简单地写

```
[HTML 中的标题标识符]
```

或

```
[HTML 中的标题标识符][]
```

或

```
[关于标题标识符的章节][HTML 中的标题标识符]
```

而不用像这样直接给出 ID：

```
[HTML 中的标题标识符](#HTML 中的标题标识符)
```

如果有多个标题具有相同文本，则对应的链接只会链接到第一个符合的标题，这时若要链接到其他符合的标题，就必须如上所述，明确指定该标题的 ID 。

与一般的指向链接类似，这些链接都不区分大小写。

在文档中直接显式定义的链接，优先级要大于隐式的标题链接。所以在下面这个例子中，“请看” 后面的链接会指向 `bar`，而不是 `#foo`：

```
# Foo

[foo]: bar

请看 [foo]
```

区块引用

Markdown 使用 email 的习惯来创建引用区块。一个引用区块可以由一或多个段落或其他区块元素（如列表或标题）组成，并且其行首均是一个 > 符号加上一个空格。(> 符号不一定要位于该行最左边，但也不能缩进超过三个空格)。

```
> 这是一段区块引用。这
> 段话有两行。
>
> 1. 这是区块引用中的一个列表。
> 2. 第二个条目。
```

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：

```
> 这是一段区块引用。这
段话有两行。

> 1. 这是区块引用中的一个列表。
2. 第二个条目。
```

在各种区块元素中，区块引用能够包含在内的是区块引用。这就是说，区块引用是可以嵌套的：

```
> 这是一段区块引用。
>
> > 区块引用内的一段区块引用。
```

如果 > 符号后面加了一个可选的空格，那么这个空格会被当作是区块引用标记符中的一部分，而不是引用内容的缩进空格。这样如果你需要在区块引用中缩进代码块，你需要在 > 后加上五个空格：

```
>      code
```

Extension: `blank_before_blockquote`

标准 Markdown 语法在区块引用之前并不需要预留空行，Pandoc 则需要（当然除去区块引用位于文档开头的情况）。这是因为 > 符号很可能完全意外地出现在行首（也许是因为自动断行）。因此除非指定 `markdown_strict` 格式，以下内容在 pandoc 中将不会产生嵌套的区块引用：

```
> 这是一段区块引用。
>> 嵌套的。
```

字面（代码）区块

缩进代码区块

一段以四个空格（或一个 tab）缩进的文本区块会被视为字面文本 (verbatim text)：换句话说，特殊字符并不会触发格式转换，所有的空格与换行符也都会被保留。例如，

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

位于每行开头的初始缩进（四个空格或一个 tab）不会被视为字面文本的一部分，因此在输出时会被移除。

注意：字面文本内的空行并不需要以四个空格开始。

围栏代码区块

Extension: `fenced_code_blocks`

除了标准的缩进代码区块，Pandoc 也支持**围栏** (*fenced*) 代码区块的语法。这种区块的第一行是三个及以上的波浪线 (~)，最后一行也都是波浪线，并且个数必须等于或大于第一行；介于这两行之间的所有文本都会被视为代码，不需额外缩进：

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

如同一般的代码区块，围栏代码区块也必须与其前后的文本以空行隔开。

如果代码本身包含了一整行的波浪线，那么只要在区块首尾处使用更长的波浪线即可：

```
~~~~~~~~~~~~~~~~
~~~~~~~~~~
包含波浪线的代码
foo = bar
~~~~~~~~~~
~~~~~~~~~~~~~~~~
```

Extension: `backtick_code_blocks`

反引号代码区块和 `fenced_code_blocks` 一样，但是使用反引号 `` ` ``，而不是波浪号 `~`。

Extension: `fenced_code_attributes`

你也可以选择性地使用以下语法，在围栏或反引号代码区块上附加一些属性：

```
~~~~ {#mycode .haskell .numberLines startFrom="100"}
qsort []     = []
qsort (x:xs) = qsort (filter (< x) xs) ++ [x] ++
               qsort (filter (>= x) xs)
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
```

这里的 `mycode` 为 ID，`haskell` 与 `numberLines` 是类名，而 `startsFrom` 则是值为 `100` 的属性。有些输出格式可以利用这些信息来作语法高亮。目前仅有 HTML 与 LaTeX 两种输出格式会使用到这些信息。如果指定的输出格式及语言类别支持语法高亮，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。（要

查找支持的编程语言列表，可在命令行输入 `pandoc --list-highlight-languages`。）反之若无支持，则上面的代码区块则会变成这样：

```
<pre id="mycode" class="haskell numberLines" startFrom="100">
  <code>
  ...
  </code>
</pre>
```

还有一种简便的方式指定代码区块的编程语言：

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

等价于：

```
``` {.haskell}
qsort [] = []
```
```

如果 `fenced_code_attributes` 扩展被禁用，而输入文本中的代码区块包含类属性，那么第一个类名会变成裸文本，不包含任何标记，输出到代码区块的第一行波浪线围栏之后。

要取消所有语法高亮，可以使用 `--no-highlight` 选项。要设置语法高亮的样式，则使用 `--highlight-style`。了解更多关于高亮的信息，请阅读下面的语法高亮小节。

行区块

Extension: `line_blocks`

行区块是一连串以竖线 (`|`) 加上一个空格开头的连续行。行与行间的分隔在输出时将以原样保留，行首的所有空格也一样会保留；如果没有使用这个标记符，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，书写时也可以将完整一行拆成多行，但后续行必须以空格作为开始。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
```

这是从 reStructuredText 借来的语法。

列表

无序列表

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

```
* 一
* 二
* 三
```

这会产生一个 “紧凑” 的列表。如果你想要一个 “宽松” 的列表，也就是说以段落格式处理每个项目的文本，那么只要在每个项目间加上空行即可：

```
* 一

* 二

* 三
```

项目符号不一定要位于行的最左端，允许它以一至三个空格作缩进。项目符号后必须跟着一个空格。

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

```
* 这是我的第一个
  列表项目。
* 然后第二个。
```

但 Markdown 也允许以下 "偷懒" 的格式：

```
* 这是我的第一个
列表项目。
* 然后第二个。
```

四空格规则

一个列表项目可以包含多个段落以及其他区块级的内容。然而，后续的段落必须接在空行之后，并且以四个空格或一个 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格），看上去会比较整齐美观：

```
  * 第一段。

    继续。

  * 第二段。包含代码区块，因此必须缩进
    八个空格。

        { code }
```

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

```
* 水果
    + 苹果
        - 麦金托什
        - 红色 好吃
```

```
    + 梨
    + 桃
* 蔬菜
    + 花椰菜
    + 叶甜菜
```

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。

```
+ 一个懒，懒的，列表
项目。

+ 再来一个；这看起来
很糟，但是符合规则。

    第二个项目的第二段
内容。
```

注意：尽管针对接续段落的“四个空格规则”是出自于官方的 Markdown syntax guide，但是作为对应参考用的 `Markdown.pl` 实现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格时，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“四个空格规则”是否适用于 **所有**位于列表项目里的区块元素；规范文档中只提及了段落与代码区块。但文档暗示了此规则适用于所有区块等级的内容（包含嵌套列表），所以 pandoc 以此为方向进行了解读与实现。

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枚举编号作开头，而不是项目符号。

在标准 Markdown 中，枚举编号是阿拉伯数字后面接着一个句点与空格。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.  一
2.  二
3.  三
```

上下两个列表的输出是相同的。

```
5.  一
7.  二
1.  三
```

Extension: `fancy_lists`

与标准 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，也可以紧跟一个右括号，抑或是句点，在这些标记符号和项目文本之间必须有一个空格。如果列表标记是大写字母接着一个句点，句点后请使用至少两个空格。[1]

`fancy_lists` 扩展同样允许 "`#`" 作为有序列表的标记符，替代数字：

```
#. 一
#. 二
```

Extension: `startnum`

Pandoc 会注意列表标记符的种类和起始编号，如果可能的话，这些信息都会在输出格式中保留。举例来说，下面的输入会产生一个从编号 9 开始，以单括号为标记的列表，底下还嵌套着一个以小写罗马数字编号的子列表：

```
 9)  九
10)  十
11)  十一
       i. 子项目一
      ii. 子项目二
     iii. 子项目三
```

当遇到不同形式的列表标记符时，pandoc 会重新开始一个新的列表。所以，以下的输入会产生三个列表：

[1] 之所以有这条规则，主要是要避免以人名缩写作为开头的段落所带来的混淆，像是

```
B. Russell 是一个英国哲学家.
```

这样的段落就不会被当作列表项目了。
这条规则并不会避免以下

```
(C) 2007 Joe Smith
```

这样的叙述被解释成列表项目。在这情形下，可以使用反斜线：

```
(C\) 2007 Joe Smith
```

```
(2) 二
(5) 三
1. 四
*  五
```

如果需要默认的有序列表标记符，可以使用 `#.`：

```
#. 一
#. 二
#. 三
```

定义列表

Extension: `definition_lists`

Pandoc 支持定义列表，使用了 PHP Markdown Extra 以及它的一些扩展。[2]

```
条目 1

:   定义 1

条目 2 带有 *行内标记格式*

:   定义 2

        { 一些代码，定义 2 的一部分}

    定义 2 的第三个段落。
```

每个专有名词条目 (term) 都必须单独存在于一行，后面可以接着一个空行，也可以省略，但一定要接上一项或多项定义内容。一项定义需由一个冒号或波浪线作开头，同时也可以前置一个或两个空格作为缩进。一个专有名词条目可以有多项定义，而每项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格或一个 tab。定义本身的内容主体（包括以冒号或波浪线开始的第一行）应该以四个空格缩进。但是，与其他 Markdown 列表类似，你可以偷个懒，在定义内的段落或区块元素中，省略除第一行外的缩进。

```
条目 1
```

[2] David Wheeler 对于 Markdown 的建议影响了我。

: 定义
包括“偷懒”的部分内容。

定义的第二段内容。

如果你在定义内容后面留下空行（如同上面的例子），那么该段定义会被当作段落处理。在某些输出格式中，这意味着成对的专有名词与定义内容间会有较大的空白间距。省略定义前的空行，即可产生一个比较紧凑的定义列表：

```
条目 1
  ~ 定义 1

条目 2
  ~ 定义 2a
  ~ 定义 2b
```

注意，定义列表中，条目间的空行是必须的。（有一个变体放宽了这个限制，但是会禁用“偷懒”式行对齐，使用 `compact_definition_lists` 扩展来激活这个功能，参见下面的Non-pandoc 扩展部分。）

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，直到文档结束。范例条目的编号不会局限於单一列表中，而是文档中所有以 `@` 为标记的项目均会依次序递增其编号，直到最后一个。举例如下：

```
(@)  我的第一个例子会被编号为 (1)。
(@)  我的第二个例子会被编号为 (2)。

各个例子的解释。

(@)  我的第三个例子会被编号为 (3)。
```

编号范例可以加上标签，并且在文档的其他地方作参照：

```
(@good)  这是一个好的 (good) 的例子。

正如 (@good) 里面表明的，...
```

标签可以是由任何英文字母、下划线或是连字符所组成的字符串。

紧凑与宽松列表

在处理某些与列表相关的 "边界情况" 时，Pandoc 与 `Markdown.pl` 使用不同的方式。考虑如下内容：

```
+   一
+   二：
    -   飞
    -   斐
    -   肥

+   三
```

Pandoc 会将以上列表转换为 "紧凑列表"（在 "一""二" 或 "三" 之间没有 <p> 标签），而 Markdown 则会在 "二" 与 "三"（但不包含 "一"）中间置入 <p> 标签，这来源于 "三" 之前的空行。Pandoc 依循着一个简单规则：如果文本后面跟着空行，那么就会被视为段落。既然 "二" 后面是跟着一个列表，而非空行，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行，那就无关紧要了。（注意：即使是设置为 `markdown_strict` 格式，Pandoc 仍是依以上方式处理。这种方式与 Markdown 官方语法规范里的描述一致，即使后者与 `Markdown.pl` 有一些分歧。）

结束一个列表

如果你在列表之后放入一个缩进的代码区块，会有什么结果？

```
-   项目一
-   项目二

    {  我的代码块  }
```

麻烦大了！此处 pandoc（与其他的 Markdown 实现一样）会将 `{  我的代码块  }` 视为 项目二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 项目二之后 “切断” 列表，你可以插入一些没有缩进、输出时也不可见的内容，例如 HTML 的注释：

```
-   项目一
-   项目二

<!-- 列表的结束 -->

    {  我的代码块  }
```

当你想要两个连续而各自独立的列表，而非一个大列表时，也可以运用同样的技巧：

```
1.  一
2.  二
3.  三

<!-- -->

1.  乌
2.  多
3.  楚
```

水平分割线

若三个及以上的 *, - 或 _ 符号（中间可以空格分隔）排列在一行中，则会产生一条水平分割线：

```
*  *  *  *

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

表格

有四种形式的表格可以使用。前三种适用于等宽字体的编辑环境，例如 Courier。第四种则不要求列之间的对齐，因此可以在比例间距字体的环境中使用。

Extension: `table_captions`

四种形式的表格都可以选择性地添加标题（见下面的例子）。标题是一个以 `Table:` (或仅是 `:`) 开头的文本段落，`Table:` 标记符在处理时会被自动去掉。标题可以放在表格的前面或后面。

Extension: `simple_tables`

简单表格看起来像这样：

```
      右      左          居中        默认
-------      ------ ----------   ----
      12     12         12            12
     123     123        123          123
       1     1           1             1
```

`Table:` 展示简单表格的语法

表头与数据行必须各自维持在同一行内。列的对齐则依照表头的文本和其底下虚线的相对位置来决定：[3]

- 如果虚线与表头文本的右侧对齐，而左侧比表头文本长，则该列为靠右对齐。
- 如果虚线与表头文本的左侧对齐，而右侧比表头文本长，则该列为靠左对齐。
- 如果虚线的两侧都比表头文本长，则该列为居中对齐。
- 如果虚线与表头文本的两侧都对齐，则会套用默认的对齐方式（在大多数情况下，这将会是靠左对齐）。

表格底下必须接一个空行，或是一行虚线后再加一个空行。如果以一行虚线结束表格，则可以不用添加表头行。例如：

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

当省略表头时，列的对齐会参考表格第一行的文本。所以在上面的表格中，各列的对齐依次会是靠右、靠左、居中以及靠右对齐。

Extension: `multiline_tables`

多行表格允许表头与单元格文本以复数行呈现（但不支持横跨多栏或纵跨多列的单元格）。以下为范例：

[3]这个方案是由 Michel Fortin 在 Markdown discussion list 的讨论中所提出。

```
-------------------------------------------------------------
    居中      默认                右 左
    表头      对齐              对齐 对齐
----------- ---- --------------- ----------------------------
    第一      行               12.0 这是一个例子，本行
                                    包含多行内容。

    第二      行                5.0 这是另一个例子。
                                    注意两行之间的
                                    空行。
-------------------------------------------------------------

Table: 本处是表格标题。它也可以
跨越多行。
```

这些看起来很像简单表格，但仍有以下差别:

- 必须以一行虚线开始，位于表头之上（除非已省略表头）。
- 必须以一行虚线结束，之后接一个空行。
- 行与行之间必须以空行隔开。

在多行表格中，表格分析器会计算各列的宽度，并在输出时尽可能维持各列在原始文档中的宽度相对比例。因此如果你觉得某列在输出时太窄，你可以在 Markdown 的源文件中加宽一点。

和简单表格一样，表头在多行表格中也可以省略:

```
----------- ---- --------------- ----------------------------
    第一      行               12.0 这是一个例子，本行
                                    包含多行内容。

    第二      行                5.0 这是另一个例子。
                                    注意两行之间的
                                    空行。
-------------------------------------------------------------

: 这是一个没有表头的多行表格。
```

多行表格可以只有一行，但该行之后必须紧跟一个空行（然后才是结束表格的一行虚线）。如果没有此空行，此表格将可能被解读成简单表格。

Extension: `grid_tables`

网格表格看起来像这样:

: 网格表格范例。

```
+---------------+---------------+--------------------+
| 水果          | 价格          | 优点               |
+===============+===============+====================+
| 香蕉          | $1.34         | - 好剥皮           |
|               |               | - 颜色亮丽         |
+---------------+---------------+--------------------+
| 橙子          | $2.10         | - 治疗坏血症       |
|               |               | - 味道好           |
+---------------+---------------+--------------------+
```

以 = 连接成的一行区分了表头与表格主体，这在无表头的表格中可以省略。网格表格的单元格可以包含任意区块元素（多个段落、代码区块、列表等等）。不支持横跨多栏或纵跨多列的单元格。网格表格可以在 Emacs table mode 下轻松创建。

像下一小节的管线表格一样，网格表格中，列的对齐方式可以通过在表头分隔符的两端放置冒号来设定:

```
+---------------+---------------+--------------------+
| 右            | 左            | 居中               |
+==============:+:==============+:==================:+
| 香蕉          | $1.34         | 好剥皮             |
+---------------+---------------+--------------------+
```

对于无表头的表格，冒号需要放在最顶上一行:

```
+--------------:+:--------------+:------------------:+
| 右            | 左            | 居中               |
+---------------+---------------+--------------------+
```

Extension: `pipe_tables`

管线表格看起来像这样:

```
| 右     | 左    | 默认     | 居中    |
|------:|:-----|---------|:------:|
|   12  |  12  |    12   |    12  |
|  123  |  123 |   123   |   123  |
|    1  |    1 |     1   |     1  |
```

: 展示管线表格的语法

这个语法与 PHP Markdown Extra 中的表格语法 相同。开始与结尾行的管线字符是可选的，但各列间则必须以管线区隔。上面范例中的冒号表明了对齐方式。表头不可以省略。要模拟一个无表头的表格，让表头的内容为空即可。

因为管线界定了各列的边界，表格并不需要一定像上面例子中那样各列对齐。所以，底下是一个完全合法（虽然丑陋）的管线表格：

```
水果| 价格
----|----:
苹果|2.05
梨|1.37
橙子|3.09
```

管线表格的单元格不能包含如段落、列表之类的区块元素，也不能包含多行文本。如果管线表格中，一行包含的可打印内容的宽度大于列的宽度（参见 `--columns`），那么单元格里的内容会根据需要自动断行，宽度由分隔行的相对宽度决定。

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

```
| 一    | 二      |
|-----+-------|
| 我的| 表格    |
| 好    | 棒！    |
```

主要的差别在于以 `+` 取代一个 `|`。其他的 orgtbl 功能并未支持。如果要指定非默认的列对齐形式，你仍然需要像前面的范例一样，在分隔行中加入冒号。

元数据区块

（译注：本节中提到的 “标题” 均指 Title，而非 Headers）

Extension: `pandoc_title_block`

如果文件以题目（Title）区块开头：

```
% 题目
% 作者（们）（以分号隔开）
% 日期
```

这部分将会作为书目信息处理，而不是一般文本。（比如单一 LaTeX 或是 HTML 输出文档的的题目，就可用这种方式呈现。）这个区块仅能包含题目，或题目与作者，或题目、作者与日期。如果你只想包含作者却不想包含题目，或是只有题目与日期而没有作者，你得利用空行：

```
%
% 作者

% 我的题目
%
% 2006 年 6 月 15 日
```

题目可包含多行文本，但接续行必须以空格开头，比如：

```
% 我的题目
  有多行内容
```

如果文档有多个作者，名单可以分列在多行，并以空格开头，或是以分号隔开，或者两种方式并用。所以，下列各种写法效果相同：

```
% 第一作者
  第二作者

% 第一作者；第二作者

% 第一作者；
  第二作者
```

日期只能写在一行之内。

所有这三个元数据 (metadata) 字段都可以包含标准的行内格式（斜体、链接、脚注等等）。

题目区块一定会被解析处理，但只有在命令行添加了 `--standalone` (`-s`) 选项时，才会影响输出内容。在输出 HTML 时，题目会出现的地方有两处：一处是文档头，即 `<head>` 区块里——这会显示在浏览器窗口的标题栏中——另一处是文档主体，即 `<body>` 区块的最前面。同时，可以通过在命令行中添加 `--title-prefix` 或 `-T` 选项来指定文档头。在文档主体里的题目会以 H1 元素呈现，并附带 "title" 类 (class)，这样就能借由 CSS 来隐藏题目或重新定义它的样式。如果以 `-T` 选项指定了文档头的内容，而源文件中却没有添加题目区块，那么命令行中指定的文本内容会成为 HTML 主体部分最上面的题目。

而 man page 的输出器会分析文档题目区块的题目行，以解析出题目、man page 章节编号，以及页眉 (header) 和页脚 (footer) 信息。题目行的第一个词会被当作文档题目，有时这一行会以括号包围的章节编号（单一数字）结束。（题目与括号之间没有空格）。在此之后的其他文本则为页脚与页眉文本。页脚与页眉文本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` 且章节编号为 1 的 man page。

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

产生的 man page 会在页脚处加上 "Pandoc User Manuals" 。

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

产生的 man page 会再在页眉处加上 "Version 4.0" 。

Extension: `yaml_metadata_block`

一个 YAML 元数据区块是一个有效的 YAML 对象，顶上一行包含 3 个连字符 (`---`)，底下一行包含三个 3 个连字符 (`---`) 或三个句点 (`...`)，以此将它与其他部分隔离开。一个 YAML 元数据区块可以出现在文档中的任何部分，但是如果它不是在文档开头，则区块之前必须有一个空行。（注意，由于 pandoc 可以将几个输入文档叠加到一起，所以你也可以将元数据单独放到一个 YAML 文件里，然后把它当作参数，与你的 Markdown 文件一起传递给 pandoc：

```
pandoc chap1.md chap2.md chap3.md metadata.yaml -s -o book.html
```

只需要保证 YAML 文件以 `---` 开头，并以 `---` 或 `...` 结束。

处理程序会从 YAML 对象的字段中获取元数据，并添加到文档中已经存在的元数据里。元数据可以包含列表和对象（可任意嵌套），但是所有的普通字符串都会被当作 Markdown 源文件解读。Pandoc 会忽略名字以下划线结尾的字段（它们可能来自于外部处理程序。）

一个文档可以包含多个元数据区块。元数据字段会根据 偏左统一原则结合起来：如果两个元数据区块试图设置同一个字段的值，那么取第一个区块的值。

使用 `pandoc -t markdown` 来创建 Markdown 文件时，只有同时加上 `-s`/`--standalone` 选项，才会生成 YAML 元数据区块。所有的元数据会集中出现在文档开头的一个单独区块中。

注意，在使用 YAML 元数据时，务必遵循转义字符规则。比如，如果一个题目中包含一个冒号（半角），那么必须为题目加上引号。管线字符 (`|`) 可以用来开始一段缩进区块，里面的内容会直接以字面字符处理，不需要转义。当字段内容包含空行时，必须选择这种形式。

```
---
title:  '这是一个题目：它包含一个半角冒号'
author:
- 第一作者
- 第二作者
tags: [空，空无]
abstract: |
  这是摘要。

  它包含两个段落。
...
```

模版变量会根据元数据自动设置。比如 `abstract` 变量在 Markdown 的 `abstract` 字段中，会变成对应的 HTML：

```
<p>这是摘要。</p>
<p>它包含两个段落。</p>
```

变量可以包含任意 YAML 结构，但是模版必须与它的结构相匹配。在默认的模版中，`author` 变量的期望值是一个简单列表或者字符串，但是可以通过修改，来支持更多复杂的结构。比如在下面的例子中，通过组合，会在作者后面添加一个 `affiliation` （附属）字段：

```
---
title: 文档题目
author:
```

```
- name: 第一作者
  affiliation: 某地某大学
- name: 第二作者
  affiliation: 克莱登大学
...
```

为了使用上面例子中结构化的作者名单，你需要一个自定义的模版：

```
$for(author)$
$if(author.name)$
$author.name$$if(author.affiliation)$ ($author.affiliation$)$endif$
$else$
$author$
$endif$
$endfor$
```

反斜线转义

Extension: `all_symbols_escapable`

除了代码区块或行内代码之外，任何标点符号或空格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，即使它通常是格式标记符号。因此，举例来说，下面的写法

```
*\*你好\**
```

输出后会得到

```
<em>*你好*</em>
```

而不是

```
<strong>你好</strong>
```

这条规则比标准的 Markdown 规则好记许多，标准规则中，只有以下字符才支持反斜线转义，:

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

（然而，如果使用了 `markdown_strict` 格式，那么就会采用标准的 Markdown 规则。）

一个反斜线之后的空格会被解读为不断行空格 (nonbreaking space)。这在 TeX 的输出中会显示为 ~，而在 HTML 与 XML 中则是 `\ ` 或 `\ `。

一个反斜线之后的换行符（即反斜线符号出现在一行的末尾）会被解读为强制换行符。这在 TeX 的输出中会显示为 `\\`，而在 HTML 里则是 `<br />`。相对于标准 Markdown 以行尾添加两个空格这种 "看不见" 的方式进行强制换行，反斜线加换行符是比较好的替代方案。

反斜线转义机制在代码上下文中不起作用。

智能型标点符号

Extension

如果指定了 `--smart` 选项，pandoc 将会输出正式印刷用的标点符号，将直引号 (straight quotes) 转换为曲引号 (curly quotes[4]) 、`---` 转为破折号 (em-dashes)，`--` 转为连接号 (en-dashes)，以及将 `...` 转为省略号。在某些缩略词之后，例如 "Mr."，会插入不断行空格 (nonbreaking spaces) 。

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{...}` 在引用文本上。

行内格式

强调

要 *emphasize* （强调[5]）某些文本，只要以 `*` 或 `_` 符号前后包住即可，像这样：

这一段文字是 `_emphasized with underscore_`（_用下划线强调_），而这

[4]译注：直引号 (straight quotes) 指的是左右两侧都一样的引号，例如我们直接在键盘上打出来的单引号或双引号 (`'`/`"`)；曲引号 (curly quotes) 则是左右两侧不同，并且视觉上看起来正好包住被引内容，中文引号即是全角曲引号 (“”)，西文印刷品中也会使用曲引号。

[5]译注：第一级强调，即使用 `*` 或 `_` 产生斜体效果，需要字体支持斜体。中文字体一般不支持斜体，所以在中文中使用单个 `*` 或 `_` 将不起作用。

一段是 `*emphasized with asterisks*`（*用星号强调*）。

重复两个 * 或 _ 符号以产生 **更强烈的强调**：

这是 `**更强烈的强调**` 以及 `__用下划线的方式实现__` 。

一个前后以空格包住，或是前面加上反斜线进行转义的 * 或 _ 符号，都不会触发强调格式：

```
This is * not emphasized *, and \* neither is this\*.
```
（这 * 不会被强调 *，这 * 也不会 *。）

Extension: `intraword_underscores`

因为 _ 字符有时会用在词组和元素标识 ID 之中，所以 pandoc 不会把被字母包住的 _ 解读为强调标记。如果需要特别强调单词中的一部分，就用 *：

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

删除线

Extension: `strikeout`

要将一段文本加上水平线作为删除效果，将该段文本前后以 ~~ 包住即可。例如，

这是 `~~被删除的内容。~~`

上标与下标

Extension: `superscript`, `subscript`

要输入上标，可以用 ^ 字符将上标文本包起来；要输入下标，可以用 ~ 字符将下标文本包起来。直接看范例，

```
H~2~O 是一种液体。 2^10^ 等于 1024。
```

如果上标或下标的文本中包含空格，那么这个空格之前必须加上反斜线，进行转义。（这是为了避免使用 ~ 和 ^ 时，意外地产生上标或下标。）所以，如果你想要让字母 P 后面跟着下标文本 'a cat'，那么就要输入 `P~a \ cat~`，而不是 `P~a cat~`。

字面文本

要让一小段文本直接以其字面形式呈现，可以用反引号将其包住：

```
`>>=` 跟 `>>` 有何区别？
```

如果字面文本中也包含了反引号，那就使用双重反引号包住：

```
这是一个字面反引号： `` ` ``。
```

（在起始双重反引号后的空格以及结束反引号前的空格都会被忽略。）

通用的规则是，字面文本以一串连续的反引号开始（反引号后的空格为可选），并以同样数目的连续反引号结束（反引号前的空格也为可选）。

要注意的是，反斜线转义字符（以及其他 Markdown 结构）在字面文本的上下文中是没有效果的：

```
这是一个反斜线后跟着一个星号： `\*`。
```

Extension: `inline_code_attributes`

与围栏代码区块一样，字面文本也可以附加属性：

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

小体大写

为了实现小体大写（small caps，小型的大写字母），你可以使用 HTML 的 span 标签：

```
<span style="font-variant:small-caps;">Small caps</span>
```

（分号是可选的，冒号后可以有空格。）这种做法在所有支持小体大写的输出格式中都可行。

另外也可以使用新添加的 `bracketed_spans` 语法来实现：

```
[Small caps]{style="font-variant:small-caps;"}
```

数学

Extension: `tex_math_dollars`

所有介于两个 `$` 字符之间的内容将会被视为 TeX 数学公式处理。开头的 `$` 右侧必须紧跟任意非空格字符，而结尾 `$` 的左侧同样也必须紧挨着非空格字符，并且结尾 `$` 的右侧不能紧跟着数字。这样一来，`$20,000 and $30,000` 这样一段话就不会被当作数学公式处理了。如果基于某些原因，必须使用字面的 `$` 符号包围其他文本时，那么可以在 `$` 前加上反斜线进行转义，这样 `$` 就不会被当作数学公式的分隔符。

TeX 数学公式会打印到所有输出格式中。至于会以什么方式编排呈现 (render)，则取决于输出格式：

Markdown, LaTeX, Emacs Org Mode, ConTeXt, ZimWiki 公式会以字面文本呈现在两个 `$` 符号之间。

reStructuredText 公式会使用此处所描述的 `:math:` 这个 “被解读文本角色” (interpreted text role) 来编排呈现。

AsciiDoc 公式会以 `latexmath:[...]` 格式编排呈现。

Texinfo 公式会在 `@math` 指令中编排呈现。

groff man 公式会去掉 `$` 后，直接以字面文本编排呈现。

MediaWiki, DokuWiki 公式会在 `<math>` 标签中编排呈现。

Textile 公式会在 `<span class="math">` 标签中编排呈现。

RTF, OpenDocument, ODT 如果可以的话，公式会以 Unicode 字符编排呈现，不然就直接使用字面字符。

Docbook 如果使用了 `--mathml` 选项，公式就会在 `inlineequation` 或 `informalequation` 标签中，使用 MathML 进行编排呈现。否则，如果可能的话，会使用 Unicode 字符编排呈现。

Docx 公式会以 OMML 数学标记的方式编排呈现。

FictionBook2 如果使用了 `--webtex` 选项，会使用 CodeCogs 或其他可用的网络服务，将公式渲染为图片，下载后嵌入电子书中。否则就会以字面文本显示。

HTML, Slidy, DZSlides, S5, EPUB 公式在 HTML 中的编排呈现方式，会依照命令行选项进行设置：

1. 默认方式是将 TeX 数学公式尽可能地以 Unicode 字符编排呈现，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，所以可以在需要时设定不同的样式，使其与周围文本内容有所不同。
2. 如果使用了 `--latexmathml` 选项，TeX 数学公式会被 `$` 或 `$$` 字符括住，并放在附带 `LaTeX` 类的 `<span>` 标签里。这段内容会被 LaTeXMathML 脚本编排为数学公式。（这个方法无法适用于所有浏览器，但在 Firefox 中是有效的。在不支持 LaTeXMathML 的浏览器中，TeX 数学公式在两个 `$` 字符中间，以字面文本呈现。）
3. 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 脚本来进行编排呈现。
4. 如果使用了 `--mimetex` 选项，mimeTeX CGI 脚本会被调用，用来生成每个 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 位于 `/cgi-bin/mimetex.cgi`。
5. 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` 文件可以使用 gladTeX 处理，这会为每个数学公式生成一个图片，并在生成的 HTML 文件中包含这些图片的链接。所以，整个处理流程如下：

   ```
   pandoc -s --gladtex myfile.txt -o myfile.htex
   gladtex -d myfile-images myfile.htex
   # 生成 myfile.html 文件和 myfile-images 文件夹，里面包含公式图片
   ```

6. 如果使用了 `--webtex` 选项，TeX 数学公式会被转换为 `<img>` 标签，并链接到一个用以转换公式为图片的外部脚本。公式将被编码为 URL 可接受的字符串，并添加到指定的 URL 后面。如果没有指定 URL，那么将会使用 CodeCogs (`https://latex.codecogs.com/png.latex?`)。
7. 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）中，或者 `\[...\]`（用于区块数学公式）中，并且放在附带 `math` 类的 `<span>` 标签之中。MathJax 脚本会将它们编排为页面上的数学公式。

原始 HTML

Extension: `raw_html`

Markdown 允许你在文档中的任何地方插入原始 HTML（或 DocBook）内容（字面文本上下文除外，因为 <, > 和 & 都会按其字面意义解读）。（技术上而言这不算扩展功能，因为标准 Markdown 本身就提供此功能，但做成扩展形式便可以在有特殊需要时关闭此功能。）

在输出至 HTML, S5, Slidy, Slideous, DZSlides, EPUB, Markdown, Emacs Org Mode 以及 Textile 格式时，原始 HTML 代码会不作修改地保留，而在其他输出格式中，原始 HTML 代码会被去掉。

Extension: `markdown_in_html_blocks`

标准 Markdown 允许你插入 HTML"区块"：所谓的 HTML 区块是指，上下各由一个空行所隔开，开始与结尾的行都以 HTML 标签开始，并且前后标签对称，形成封闭区块。在这个区块中，任何内容都会当作是 HTML 来解析，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；但默认情况下，pandoc 会以 Markdown 语法解读 HTML 标签区块中的内容。比如 pandoc 会将底下这段

```
<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` 则保留原样。

这个规则只有一个例外：那就是介于 `<script>` 与 `<style>` 标签之间的文本都不会被当作 Markdown 解读。

这种与标准 Markdown 的差别，会让 Markdown 与 HTML 区块元素的混合变得更加容易。比方说，一段 Markdown 文本可以用 `<div>` 标签将其前后包住，而里面的文本仍然会以 Markdown 方式解读。

原始 TeX

Extension: `raw_tex`

除了 HTML 之外，pandoc 也接受文档中嵌入原始 LaTeX, TeX 以及 ConTeXt 代码。行内 TeX 指令会按原样输出至 LaTeX 与 ConTeXt 格式中。所以，举例来说，你可以使用 LaTeX 来插入 BibTeX 的引用文献：

```
这个结果在文献 \cite{jones.1967} 中有过证明。
```

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}
```

位于 `begin` 与 `end` 标签之间的内容，都会被当作是原始 LaTeX 文本解读，而不会被视为 Markdown。

除了输出至 Markdown，LaTeX，Emacs Org Mode 及 ConTeXt 四种格式，其他格式中，行内 LaTeX 会被忽略。

LaTeX 宏

Extension: `latex_macros`

当输出格式不是 LaTeX 时，pandoc 会分析 LaTeX 的 `\newcommand` 和 `\renewcommand` 定义，并将其产生的宏应用到所有 LaTeX 数学公式中。所以举例来说，下列指令对所有的输出格式均有作用，而不仅是 LaTeX：

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

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

而输出格式是 LaTeX 时，`\newcommand` 定义会按原样保留。

链接

Markdown 允许以如下方式指定链接。

自动链接

如果你用尖括号将一段 URL 或是 email 地址包起来，它会自动转换成链接：

```
<http://google.com>
<sam@green.eggs.ham>
```

行内链接

一个行内链接包含位于方括号中的链接文本和方括号后以圆括号包起来的 URL。（你可以选择性地在 URL 后面加入链接标题，标题文本要放在引号之中。）

```
这是一个[行内链接](/url)，而这是一个[带标题的链接](http://fsf.org "点击这里以获得快乐！").
```

方括号与圆括号之间不能有空格。链接文本可以包含格式（例如强调），但链接标题不行。

行内链接中的 email 地址不会自动识别为链接，所以必须在地址前加上 `mailto`：

```
[给我发邮件！](mailto:sam@green.eggs.ham)
```

指向链接

一个 *explicit* (明确）的指向链接包含两个部分，链接本身以及链接定义，其中链接定义可以放在文档的任何地方（不论是放在链接文本之前或之后）。

链接本身是由两对方括号所组成，第一对方括号中为链接文本，第二对中为链接标签。（在两对方括号之间可以有空格。）链接定义则是以方括号框住的链接标签作开头，后面跟着一个冒号和一个空格，然后接着一个 URL，最后可以选择性地（在一个空格之后）加入由引号或是圆括号包住的链接标题。链接标签中不允许出现可解读为引用 (citation) 的文本（假设启用了 `citations` 扩展）：在解析时引用的优先级高于链接标签。

以下是一些范例:

```
[我的链接标签一]: /foo/bar.html  "我的标题，可选"
[我的链接标签二]: /foo
[我的链接标签三]: http://fsf.org (自由软件协会)
[我的链接标签四]: /bar#special  '单引号中的标题'
```

链接的 URL 也可以选择性地以尖括号包住:

```
[我的链接标签五]: <http://foo.bar.baz>
```

链接标题可以放在第二行:

```
[我的链接标签三]: http://fsf.org
  "自由软件协会"
```

需注意链接标签并不区分大小写。所以下面的例子会创建合法的链接:

```
这是 [我的链接][FOO]

[Foo]: /bar/baz
```

在一个 *implicit* (隐性) 的指向链接中，第二组方括号的内容是空的:

```
参见[我的网站][]。

[我的网站]: http://foo.bar.baz
```

注意：在 `Markdown.pl` 以及大多数其他 Markdown 实现中，指向链接的定义不能存在于嵌套结构中，例如列表项目或是区块引用。Pandoc 解除了这个看起来很武断的限制。所以下面的语法在几乎所有其他实现中都无效，但在 pandoc 中可以正确处理:

```
> 我的区块[引用]。
>
> [引用]: /foo
```

Extension: `shortcut_reference_links`

在一个 *shorcut* (简略) 的指向链接中，第二对方括号甚至可以全部省略：

```
参见[我的网站]。

[我的网站]: http://foo.bar.baz
```

内部链接

要链接到同一份文档的其他章节，可使用自动产生的 ID 标识符（参见标题标识符 一节后半部分）。例如：

```
参见[介绍](#介绍)。
```

或是

```
参见[介绍]。

[介绍]: #介绍
```

内部链接目前支持的格式有 HTML（包括 HTML slide shows 与 EPUB）、LaTeX 以及 ConTeXt。

图片

在链接语法的前面加上一个 ! 就是图片的语法了。链接文本将会作为图片的 alt 文本：

```
![la lune](lalune.jpg "登月之旅")

![电影胶卷]

[电影胶卷]: movie.gif
```

Extension: `implicit_figures`

一个图片若自身单独存在一个段落中，那么将会以附图片说明 (caption) 的图表 (figure) 呈现。[6]（在 LaTeX 中，会使用图表环境 (figure environment)；在 HTML 中，图片会被放在具有 `figure` 类的 `div` 元素中，并会附上一个具有 `caption` 类的 `p` 元素。）图片的 alt 文本同时也会用来作为图片说明。

```
![这是图片说明](/url/of/image.png)
```

如果你只想要个普通的行内图片，那么只需要保证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：

```
![这张图片不会成为一个图表](/url/of/image.png)\
```

Extension: `link_attributes`

链接和图片都可以设置属性：

```
一张行内 ![图片](foo.jpg){#id .class width=30 height=20px}
和一张带有属性的 ![指向][ref] 图片.

[ref]: foo.jpg "可选的标题" {#id .class key=val key2="val 2"}
```

（当只用 `#id` 和 `.class` 时，此语法与 PHP Markdown Extra 兼容。）

对于 HTML 和 EPUB 来说，除了 `width` 和 `height`，所有属性（包括 `scrset` 和 `sizes`）都会按原样输出。而输出到其他格式时，输出格式不支持的属性将被忽略。

`width` 和 `height` 属性会被特别处理。当这两个属性的值不带单位时，pandoc 假设单位是像素 (pixels)。但是这些单位都可以使用：`px`, `cm`, `mm`, `in`, `inch` 和 `%`。数字和单位之间，不能有任何空格。例如：

```
![](file.jpg){ width=50% }
```

- 在以纸张页面为基础的格式中，比如 LaTeX，宽度和高度会转化为英寸。HTML 形式的输出格式中，它们会转化为像素。可以使用 `--dpi` 选项来指定每英寸的像素数，默认值是 96dpi。
- 通常`%` 单位是相对某些可供利用的空间来说的，比如上面的例子会被转化成 `<img href="file.jpg" style="width: 50%;" />` (HTML),`\includegraphics[width=0.5\textwidth]{file.jpg}` (LaTeX)，或者 `\externalfigure[file.jpg][width=0.5\textwidth]` (ConTeXt)。

[6]这项功能尚未在 RTF, OpenDocument 或 ODT 格式上实现。在这些格式中，你会得到一个段落，只包含图片，而没有图片说明。

- 输出格式中，有些有 “类”(class) 的概念 (ConTeXt)，有些有 ID (identifier) 的概念，有些两种都有 (HTML)。
- 当没有指定 `width` 和 `height` 属性时，折中的办法是读取图片的分辨率和嵌在图片文件中的 dpi 元数据。

Spans

Extension: `bracketed_spans`

由方括号包住的一串字符，通常用来创建一个链接，但如果它紧跟着属性的定义，那么它会被处理成带有属性的 span 元素。

```
[这是 *some text*]{.class key="val"}
```

脚注

Extension: `footnotes`

Pandoc's Markdown 支持脚注功能，使用以下的语法:

```
这是一个脚注链接，[^1] 然后另一个。[^longnote]

[^1]: 这是一段脚注。

[^longnote]: 这是一段包含多个区块的脚注。

    后面接续的段落必须缩进，以表明它属于
前面的脚注。

        { 一些代码 code }

    整段都可以缩进，也可以只缩进段落的第一
    行。这样多段落脚注的用法，就和多段落的列表项
    目一样。

这一段将不会成为脚注的一部分，因为它
没有缩进。
```

脚注链接的 ID 不得包含空格、tabs 或换行符。这些 ID 只用于将脚注链接和脚注内容关联起来；在输出时，脚注将会依次递增编号。

脚注本身不需要放在文档的最后面。它们可以放在文档里的任何地方，但不能被放入区块元素（列表、区块引用、表格等）之中。每一个脚注都必须使用前后的空行，将其与周围内容（包括其他脚注）隔离开。

Extension: `inline_notes`

Pandoc 也支持行内脚注（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：

```
这是一个行内脚注。^[行内脚注更容易书写，因为你不需要特意选择一
个标识符，然后再到下面去输入脚注内容。]
```

行内脚注与普通脚注可以自由混合使用。

引用

Extension: `citations`

使用外部过滤器 (filter) `pandoc-citeproc`，pandoc 能够以数种样式自动生成引用与参考文献。基本的用法如下：

```
pandoc --filter pandoc-citeproc myinput.txt
```

为了使用这项功能，你需要在命令行中指定一个参考文献文件，即在 YAML 元数据区块中，使用 `bibliography` 元数据字段，或者直接在命令行中加入 `--bibliography` 参数。如果你想使用多个参考文献文件，你可以提供多个 `--bibliography` 参数，或者将 `bibliography` 设置成 YAML 数组。可用的参考文献格式：

格式	文件扩展名
BibLaTeX	.bib
BibTeX	.bibtex
Copac	.copac
CSL JSON	.json
CSL YAML	.yaml
EndNote	.enl
EndNote XML	.xml

格式	文件扩展名
ISI	.wos
MEDLINE	.medline
MODS	.mods
RIS	.ris

注意，扩展名 `.bib` 同时适用于 BibTeX 与 BibLaTeX 文件，不过你可以使用 `.bibtex` 来强制指定 BibTeX。

注意，命令 `pandoc-citeproc --bib2json` 和 `pandoc-citeproc --bib2yaml` 可以将任何其他支持的格式转换成 `.json` 和 `.yaml` 文件。

各种字段内标记语言 (in-field markup)：在 BibTeX 和 BibLaTeX 数据库中，pandoc-citeproc 会解析 LaTeX 标记语言的子集；在 CSL YAML 数据库中，是 pandoc Markdown；在 CSL JSON 数据库中，是与 HTML 近似的标记语言：

```
<i>...</i>
```
斜体
```
<b>...</b>
```
粗体
```
<span style="font-variant:small-caps;">...</span> **or**
<sc>...</sc>
```
小体大写 (small capitals)
```
<sub>...</sub>
```
下标
```
<sup>...</sup>
```
上标
```
<span class="nocase">...</span>
```
阻止短语的大小写被转换成标题形式

`pandoc-citeproc -j` 和 `-y` 命令会尽可能完整地在 CSL JSON 和 CSL YAML 两种格式间互相转换。

除了使用 `--bibliography` 命令行选项或者 YAML 元数据中的 `bibliography` 字段来指定参考文献文件，你也可以在文档本身的 YAML 元数据区块中，将引用数据直接写在 `references` 字段后。这个字段需要包含一个 YAML 数组，并在里面对参考文献信息进行合适的编排，比如：

```
---
references:
- type: article-journal
  id: WatsonCrick1953
```

```
  author:
  - family: Watson
    given: J. D.
  - family: Crick
    given: F. H. C.
  issued:
    date-parts:
    - - 1953
      - 4
      - 25
  title: 'Molecular structure of nucleic acids: a structure for deoxyribose
    nucleic acid'
  title-short: Molecular structure of nucleic acids
  container-title: Nature
  volume: 171
  issue: 4356
  page: 737-738
  DOI: 10.1038/171737a0
  URL: http://www.nature.com/nature/journal/v171/n4356/abs/171737a0.html
  language: en-GB
...
```

(`pandoc-citeproc --bib2yaml` 命令可以从任何支持的参考文献文件格式中，生成这样的 YAML 数组。)

引用和参考文献条目可以使用任何引用样式语言支持的样式，具体的支持列表可见 Zotero 样式仓库；可用 `--csl` 选项或者 `cls` 元数据字段来指定这些文件。默认情况下，`pandoc-citeproc` 会使用芝加哥样式手册中的 "作者－日期" 格式。CSL 项目提供了更多信息，帮助你寻找和编辑样式。

如果需要让你的引用超链接到相应的参考文献条目，在你的 YAML 元数据中加上 `link-citations: true`。

引用信息放在方括号中，以分号隔开。每一条引用都会有个索引键 (key)，由 `@` 加上数据库中的引用 ID 组成，并且可以选择性地包含前缀、页码或章节定位符以及后缀。引用的索引键必须以字母，数字，或 `_` 开头，并且可以包含字母，数字，`_` 和内部标点符号 (`:.#$%&-+?<>~/`)。以下是一些范例：

等之等之 `[see @doe99, pp. 33-35; also @smith04, ch. 1]`。

等之等之 `[@doe99, pp. 33-35, 38-39 and *passim*]`。

等之等之 `[@smith04; @doe99]`。

`pandoc-citeproc` 可以检测 CSL 区域环境 (locale) 文件中的定位符条目，简写和非简写形式都可行。在 `en-`

US 区域环境中，定位符可以使用单数或复数，比如 book, bk./bks.; chapter, chap./chaps.; column, col./cols.; figure, fig./figs.; folio, fol./fols.; number, no./nos.; line, l./ll.; note, n./nn.; opus, op./opp.; page, p./pp.; paragraph, para./paras.; part, pt./pts.; section, sec./secs.; sub verbo, s.v./s.vv.; verse, v./vv.; volume, vol./vols.; ¶/¶¶; §/§§。如果数字后没有加上定位符后缀，那么将会默认使用 "page"。

在 @ 前面添加减号 (-) 将会避免作者名字在引用中出现。这可用于已经提及作者名字的场合：

```
Smith 说了等之等之 [-@smith04]。
```

你也可以在文本中直接插入引用信息，方式如下：

```
@smith04 里说到等之等之。

@smith04 [p. 33] 说到等之等之。
```

如果样式需要产生一份引用作品的列表，这份列表会被放在文档的最后面。一般而言，你需要以一个适当的标题结束你的文档：

```
最后一段...

# 参考文献
```

如此一来参考文献就会被放在这个标题后面了。注意，这个标题会自动加上 .unnumbered 类，所以这一部分不会被编号。

如果你并没有在文章主体中引用一个条目，但是却想在参考文献中列出它，你可以定义一个虚设的 nocite 元数据字段，然后把引用放在这里：

```
---
nocite: |
  @item1, @item2
...

@item3
```

在这个例子中，文档中只会包含 item3 的引用，但是参考文献中会包含 item1，item2 和 item3 的条目。

如果使用通配符，则可以创建一个参考文献列表，包含所有的引用，不论它们有没有在文档中出现。

```
---
nocite: |
  @*
...
```

当输出 LaTeX 或 PDF 文档时，你也可以使用 `natbib` 或 `biblatex` 来编排参考文献。具体做法是，在运行 `pandoc` 命令时，像上面提到的那样指定参考文献文件，然后加上 `--natbib` 或则 `--biblatex` 参数。记住，必须使用对应格式的参考文献文件（BibTeX 或 BibLaTeX）。

请打开 pandoc-citeproc 手册页面获得更多信息。

Non-pandoc 扩展

以下的 Markdown 语法扩展在 pandoc 中默认是关闭的，但是可以通过添加 `+EXTENSION` 到格式名称中，来启用这些扩展，`EXTENSION` 就是扩展的名字。比如 `markdown+hard_line_breaks` 表示带有强制换行符的 Markdown。

Extension: `angle_brackets_escapable`

允许对 < 和 > 进行反斜线转义，GitHub 风格的 Markdown 支持这种做法，但原始的 Markdown 不支持。Pandoc 默认的 `all_symbols_escapable` 扩展已经隐含了这个功能。

Extension: `list_without_preceding_blankline`

允许段落之后立即接上一个列表，不需要在中间插入空行。

Extension: `hard_line_breaks`

将导致段落内部的所有换行符都被解读成强制换行符，而不是空格。

Extension: `ignore_line_breaks`

将导致段落内部的换行符被忽略，而不是解读成空格或强制换行符。这个选项是为东亚语言准备的，因为这些语言中词语之间没有空格，但是文本会通过断行提高可读性。

Extension: `east_asian_line_breaks`

当段落内部的换行符出现在两个东亚语言字符宽度的字符之间时，这个扩展将导致这些换行符被忽略，而不是被解读成空格或强制换行符。当文本中混合了东亚语言字符宽度的字符和其他语言字符时，这个扩展要优于 `ignore_line_breaks`。

Extension: `emoji`

将文本表情，比如 `:smile:`，解析成 Unicode 表情符号。

Extension: `tex_math_single_backslash`

将导致位于 `\(` 和 `\)` 之间的所有内容被解读为行内 TeX 数学公式，而 `\[` 和 `\]` 之间的所有内容将被解读为显示用 TeX 数学公式（display TeX math）。注意：这个扩展的缺点是会使 `(` 和 `[` 的转义失效。

Extension: `tex_math_double_backslash`

将导致位于 `\\(` 和 `\\)` 之间的所有内容被解读为行内 TeX 数学公式，而 `\\[` 和 `\\]` 之间的所有内容将被解读为显示用 TeX 数学公式（display TeX math）。

Extension: `markdown_attribute`

Pandoc 默认将区块级别标签内的文本解读为 Markdown。这个扩展会改变这种行为，只有区块标签带有属性 `markdown=1` 时，内部的文本才会解读为 Markdown。

Extension: `mmd_title_block`

允许在文档开头使用 MultiMarkdown 风格的题目区块，比如：

```
Title:   我的题目
Author:  杜约翰
Date:    September 1, 2008
Comment: 这是 mmd 题目区块的例子，有一个字段
         的内容跨越多行。
```

更多细节请查看 MultiMarkdown 的文档。如果 `pandoc_title_block` 或者 `yaml_metadata_block` 被启用，那么它们的优先级高于 `mmd_title_block`。

Extension: `abbreviations`

用于解析 PHP Markdown Extra 中的缩略词索引 (keys)，例如

```
*[HTML]: Hypertext Markup Language
```

注意，pandoc 文档模型不支持缩略词，所以如果启用了这个扩展，形如上面例子中的缩略词键值对将被直接跳过（而不是解读为一个段落）。

Extension: `autolinks_bare_uris`

将所有的绝对 URI （统一资源定位符）变成链接，即使它没有被尖括号 `<...>` 包围。

Extension: `ascii_identifiers`

将 `auto_identifiers` 扩展生成的标识符 ID 变成纯粹的 ASCII 字符。带有音调的拉丁字母将被去掉音调，并且非拉丁字母会被忽略。

Extension: `mdd_link_attributes`

解析 multimarkdown 样式中链接和图片的属性键值对 (key-value)。请勿将这个扩展与 `link_attributes` 混淆。

```
这是一个指向![图片][ref]，带有 multimarkdown 格式的属性。

[ref]: http://path.to/image "图片标题" width=20px height=30px
       id=myId class="myClass1 myClass2"
```

Extension: `mmd_header_identifiers`

解析 multimarkdown 样式的标题标识符（位于方括号之中，在标题之后，但是在 ATX 标题末尾的 `#` 之前）。

Extension: `compact_definition_lists`

激活 pandoc 1.12.x 和更早版本中的定义列表语法。这个语法与上面提到的定义列表有这几方面的不同：

- 定义列表中的相邻条目之间不需要空行。
- 省略相邻条目间的空行，可以得到 "紧凑" 的列表；条目和其定义之间的空行没有任何实际影响。

- 不支持段落的偷懒式断行：整个定义必须缩进四个空格。[7]

Markdown 变体

除了 pandoc 扩展的 Markdown，也支持以下几种 Markdown 变体：

`markdown_phpextra` (**PHP Markdown Extra**) `footnotes`, `pipe_tables`, `raw_html`, `markdown_attribute`, `fenced_code_blocks`, `definition_lists`, `intraword_underscores`, `header_attributes`, `link_attributes`, `abbreviations`, `shortcut_reference_links`.

`markdown_github` (**GitHub-Flavored Markdown**) `pipe_tables`, `raw_html`, `fenced_code_blocks`, `auto_identifiers`, `ascii_identifiers`, `backtick_code_blocks`, `autolink_bare_uris`, `intraword_underscores`, `strikeout`, `hard_line_breaks`, `emoji`, `shortcut_reference_links`, `angle_brackets_escapable`.

`markdown_mmd` (**MultiMarkdown**) `pipe_tables`, `raw_html`, `markdown_attribute`, `mmd_link_attributes`, `tex_math_double_backslash`, `intraword_underscores`, `mmd_title_block`, `footnotes`, `definition_lists`, `all_symbols_escapable`, `implicit_header_references`, `auto_identifiers`, `mmd_header_identifiers`, `shortcut_reference_links`.

`markdown_strict` (**Markdown.pl**) `raw_html`

能够在非 Markdown 文件中使用的扩展

上面讨论的扩展中，有一些可以在非 Markdown 文档中使用：

- `auto_identifiers` 可以在 `latex`，`rst`，`mediawiki`，和 `textile` 输入文档中使用（默认）。
- `tex_math_dollars`，`tex_math_single_backslash`，`tex_math_double_backslash` 可以在 `html` 输入文档中使用。（举例来说，在读取使用 MathJax 格式化的网页文档时非常方便。）

[7] 为了理解为何放宽条目间的空行要求后，偷懒式断行将不可行，可以考虑以下的例子：

```
bar
:    definition
foo
:    definition
```

这是一个列表条目 “bar”，带有两个定义，其中第一个定义是偷懒式断行，还是说，是两个列表条目 “bar” 和 “foo”？为了消除歧义，我们必须作出一种选择，要么禁止偷懒式断行，要么在列表条目中插入空行。

本简体中文译本是基于曾于修 (Tzen Yuxio) 的繁体译本。