Pandoc’s Markdown 語法中文翻譯

by John MacFarlane Translated by Tzeng Yuxio

Contents

前言

這份文件是 Pandoc 版本 Markdown 語法的中文翻譯。Pandoc 本身是由 John MacFarlane 所開發的文件轉換工具，可以在 HTML, Markdown, PDF, TeX...等等格式之間進行轉換。有許多喜歡純文字編輯的人，利用 Pandoc 來進行論文的撰寫或投影片製作。但除了轉換的功能外，Pandoc 所定義的 Markdown 擴充語法也是這套工具的一大亮點，在 Pandoc 的官方使用說明文件中，光是其針對 Markdown 格式的擴充就佔了整整一半左右的篇幅。

本文件翻譯自 Pandoc – Pandoc User's Guide 中的 "Pandoc's markdown" 一節。你可以看看這份文件的原始檔、產生文件所使用的 HTML 範本，以及轉換時的命令參數。

以下翻譯開始。

Pandoc's markdown

與 John Gruber 的原始 markdown 相比，Pandoc 版本的 markdown 在語法上有額外的擴充與些許的修正。這份文件解釋了這些語法，並指出其與原始 markdown 的差異所在。除非特別提到，不然這些差異均可藉由使用 `markdown_strict` 而非 `markdown` 的格式來關閉。單獨一項擴充也可透過 `+EXTENSION` 或 `-EXTENSION` 的方式來開啟或關閉。例如，`markdown_strict+footnotes` 表示加上腳註擴充的原始 markdown，而 `markdown-footnotes-pipe_tables` 則是拿掉了腳註與管線表格擴充的 pandoc markdown。

哲學

Markdown 是針對易於書寫與閱讀的目標而設計的，特別是在易於閱讀這點上尤為重要：

> 一份 Markdown 格式的文件應該要能以純文字形式直接發表，並且一眼看過去不存在任何標記用的標籤或格式指令。John Gruber

這項原則同樣也是 pandoc 在制訂表格、腳註以及其他擴充的語法時，所依循的規範。

然而，pandoc 的目標與原始 markdown 的最初目標有著方向性的不同。在 markdown 原本的設計中，HTML 是其主要輸出對象；然而 pandoc 則是針對多種輸出格式而設計。因此，雖然 pandoc 同樣也允許直接嵌入 HTML 標

籤，但並不鼓勵這樣的作法，取而代之的是 pandoc 提供了許多非 HTML 的方式，來讓使用者輸入像是定義清單、表格、數學公式以及腳註等諸如此類的重要文件元素。

段落

一個段落指的是一行以上的文字，跟在一行以上的空白行之後。換行字元會被當作是空白處理，因此你可以依自己喜好排列段落文字。如果你需要強制換行，在行尾放上兩個以上的空白字元即可。

Extension: **escaped_line_breaks**

一個反斜線後跟著一個換行字元，同樣也有強制換行的效果。

標題

有兩種不同形式的標題語法，Setext 以及 atx。

Setext 風格標題

Setext 風格的標題是由一行文字底下接著一行 = 符號（用於一階標題）或 - 符號（用於二階標題）所構成：

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

A level-two header
------------------
```

標題的文字可以包含行內格式，例如強調（見下方行內格式 一節）。

Atx 風格標題

Atx 風格的標題是由一到六個 # 符號以及一行文字所組成，你可以在文字後面加上任意數量的 # 符號。由行首起算的 # 符號數量決定了標題的階層：

```
## A level-two header

### A level-three header ###
```

如同 setext 風格標題，這裡的標題文字同樣可包含行內格式：

```
# A level-one header with a [link](/url) and *emphasis*
```

Extension: **blank_before_header**

原始 markdown 語法在標題之前並不需要預留空白行。Pandoc 則需要（除非標題位於文件最開始的地方）。這是因為以 # 符號開頭的情況在一般文字段落中相當常見，這會導致非預期的標題。例如下面的例子：

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

HTML, LaTeX 與 ConTeXt 的標題識別符

Extension: **header_attributes**

在標題文字所在行的行尾，可以使用以下語法為標題加上屬性：

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

雖然這個語法也包含加入類別 (class) 以及鍵 / 值形式的屬性 (attribute)，但目前只有識別符 (identifier/ID) 在輸出時有實際作用（且只在部分格式的輸出，包括：HTML, LaTeX, ConTeXt, Textile, AsciiDoc）。舉例來說，下面是將標題加上 `foo` 識別符的幾種方法：

```
# My header {#foo}

## My header ##    {#foo}

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

(此語法與 PHP Markdown Extra 相容。)

具有 `unnumbered` 類別的標題將不會被編號，即使 `--number-sections` 的選項是開啟的。單一連字符號 (`-`) 等同於 `.unnumbered`，且更適用於非英文文件中。因此，

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

與下面這行是等價的

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

Extension: **auto_identifiers**

沒有明確指定 ID（識別符）的標題將會依據其標題文字，自動指派一個獨一無二的 ID。由標題文字推導 ID 的規則如下：

- 移除所有格式，連結等。
- 移除所有標點符號，除了底線、連字符號與句號。
- 以連字符號取代所有空白與換行字元。
- 將所有英文字母轉為小寫。
- 移除第一個字元前的所有內容（ID 不能以數字或標點符號開頭）。
- 如果剩下為空字串，則使用 `section` 作為 ID。

以下是一些範例，

Header	Identifier
Header identifiers in HTML	`header-identifiers-in-html`
Dogs?—in my house?	`dogs--in-my-house`
[HTML], [S5], or [RTF]?	`html-s5-or-rtf`

Header	Identifier
3. Applications	`applications`
33	`section`

在大多數情況下，這些規則應該讓人能夠直接從標題文字推導出 ID。唯一的例外是當有多個標題具有同樣文字的情況；在這情況下，第一個標題的 ID 仍舊是透過以上規則推導而得；第二個則是在同樣 ID 後加上 `-1`；第三個加上 `-2`；以此類推。

在開啟 `--toc|--table-of-contents` 的選項時，這些 ID 是用來產生目錄 (Table of Contents) 所需的頁面連結。此外，這些 ID 也提供了一個簡便的方式來輸入跳到指定章節的連結。一個以 ID 產生的連結，其使用的語法看起來就像下面的例子：

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

然而要注意的一點是，只有在以 HTML、LaTeX 與 ConTeXt 格式輸出時，才能以這種方式產生對應的章節連結。

如果指定了 `--section-divs` 選項，則每一個小節都會以 `div` 標籤包住（或是 `section` 標籤，如果有指定 `--html5` 選項的話），並且 ID 會被附加在用來包住小節的 `<div>`（或是 `<section>`）標籤，而非附加在標題上。這使得整個小節都可以透過 javascript 來操作，或是採用不同的 CSS 設定。

Extension: **`implicit_header_references`**

Pandoc 假設每個標題都定義了其參考連結，因此，相較於以下的連結語法

```
[header identifiers](#header-identifiers-in-html)
```

你也可以單純只寫

```
[header identifiers]
```

或

```
[header identifiers][]
```

或

```
[the section on header identifiers][header identifiers]
```

如果有多個標題具有同樣文字，對應的參考只會連結到第一個符合的標題，這時若要連結到其他符合的標題，就必須以先前提到的方式，明確指定連結到該標題的 ID。

與其他一般參考連結不同的是，這些參考連結是大小寫有別的。

注意：如果你有明確定義了任何一個標題的標示符，那麼選項 `implicit_header_references` 就沒有作用。

區塊引言

Markdown 使用 email 的習慣來建立引言區塊。一個引言區塊可以由一或多個段落或其他的區塊元素（如清單或標題）組成，並且其行首均是由一個 > 符號加上一個空白作為開頭。(> 符號不一定要位在該行最左邊，但

也不能縮進超過三個空白)。

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

有一個「偷懶」的形式：你只需要在引言區塊的第一行行首輸入 > 即可，後面的行首可以省略符號：

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

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

由於區塊引言可包含其他區塊元素，而區塊引言本身也是區塊元素，所以，引言是可以嵌套入其他引言的。

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

Extension: **`blank_before_blockquote`**

原始 markdown 語法在區塊引言之前並不需要預留空白行。Pandoc 則需要（除非區塊引言位於文件最開始的地方)。這是因為以 > 符號開頭的情況在一般文字段落中相當常見（也許由於斷行所致)，這會導致非預期的格式。因此，除非是指定為 `markdown_strict` 格式，不然以下的語法在 pandoc 中將不會產生出嵌套區塊引言：

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

字面（代碼）區塊

縮進代碼區塊

一段以四個空白（或一個 tab）縮進的文字區塊會被視為字面區塊 (Verbatim Block)：換句話說，特殊字元並不會轉換為任何格式，單純只以字面形式呈現，而所有的空白與換行也都會被保留。例如，

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

位於行首的縮排（四個空白或一個 tab）並不會被視為字面區塊的一部分，因此在輸出時會被移除掉。

注意：在字面文字之間的空白行並不需要也以四個空白字元做開頭。

圍欄代碼區塊

Extension: `fenced_code_blocks`

除了標準的縮進代碼區塊外，Pandoc 也支援了**圍欄** (fenced) 代碼區塊的語法。這區塊需以包含三個以上波浪線 (~) 或反引號 (`) 的一行作為開始，並以同樣符號且至少同樣長度的一行作為結束。所有介於開始與結束之間的文字行都會視為代碼。不需要額外的縮進：

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

如同一般的代碼區塊，圍欄代碼區塊與其前後的文字之間必須以空白行作間隔。

如果代碼本身也包含了一整行的波浪線或反引號，那麼只要在區塊首尾處使用更長的波浪線或反引號即可：

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

你也可以選擇性地使用以下語法附加屬性到代碼區塊上：

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

這裡的 `mycode` 為 ID，`haskell` 與 `numberLines` 是類別，而 `startsFrom` 則是值為 `100` 的屬性。有些輸出格式可以利用這些資訊來作語法高亮。目前有使用到這些資訊的輸出格式僅有 HTML 與 LaTeX。如果指定的輸出格式及語言類別有支援語法高亮，那麼上面那段代碼區塊將會以高亮並帶有行號的方式呈現。（要查詢支援的程式語言清單，可在命令列輸入 `pandoc --version`。）反之若無支援，則上面那段代碼區塊則會以下面的形式呈現：

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

下面這個是針對代碼區塊只有指定程式語言屬性的簡便形式：

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

這與下面這行的效果是相同的：

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

要取消所有語法高亮，使用 `--no-highlight` 選項。要設定語法高亮的配色，則使用 `--highlight-style`。

## 行區塊

**Extension: `line_blocks`**

行區塊是一連串以豎線 (`|`) 加上一個空格所構成的連續行。行與行間的區隔在輸出時將會以原樣保留，行首的空白字元數目也一樣會被保留；反之，這些行將會以 markdown 的格式處理。這個語法在輸入詩句或地址時很有幫助。

```
| 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
```

如果有需要的話，書寫時也可以將完整一行拆成多行，但後續行必須以空白作為開始。下面範例的前兩行在輸出時會被視為一整行：

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

這是從 reStructuredText 借來的語法。

## 清單

### 無序清單

無序清單是以項目符號作列舉的清單。每條項目都以項目符號 (`*`, `+` 或 `-`) 作開頭。下面是個簡單的例子：

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

這會產生一個「緊湊」清單。如果你想要一個「寬鬆」清單，也就是說以段落格式處理每個項目內的文字內容，那麼只要在每個項目間加上空白行即可：

```
* one

* two

* three
```

項目符號不能直接從行首最左邊處輸入，而必須以一至三個空白字元作縮進。項目符號後必須跟著一個空白字元。

清單項目中的接續行，若與該項目的第一行文字對齊（在項目符號之後），看上去會較為美觀：

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

但 markdown 也允許以下「偷懶」的格式：

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

**四個空白規則**

一個清單項目可以包含多個段落以及其他區塊等級的內容。然而，後續的段落必須接在空白行之後，並且以四個空白或一個 tab 作縮進。因此，如果項目裡第一個段落與後面段落對齊的話（也就是項目符號前置入兩個空白），看上去會比較整齊美觀：

```
  * First paragraph.

    Continued.

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

        { code }
```

清單項目也可以包含其他清單。在這情況下前置的空白行是可有可無的。嵌套清單必須以四個空白或一個 tab 作縮進：

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

上一節提到，markdown 允許你以「偷懶」的方式書寫，項目的接續行可以不和第一行對齊。不過，如果一個清單項目中包含了多個段落或是其他區塊元素，那麼每個元素的第一行都必須縮進對齊。

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

+ Another one; this looks
bad but is legal.

    Second paragraph of second
list item.
```

**注意：**儘管針對接續段落的「四個空白規則」是出自於官方的 markdown syntax guide，但是作為對應參考用的 `Markdown.pl` 實作版本中並未遵循此一規則。所以當輸入時若接續段落的縮進少於四個空白時，pandoc 所輸出的結果會與 `Markdown.pl` 的輸出有所出入。

在 markdown syntax guide 中並未明確表示「四個空白規則」是否一體適用於 **所有**位於清單項目裡的區塊元素上；規範文件中只提及了段落與代碼區塊。但文件暗示了此規則適用於所有區塊等級的內容（包含嵌套清單），並且 pandoc 以此方向進行解讀與實作。

### 有序清單

有序清單與無序清單相類似，唯一的差別在於清單項目是以列舉編號作開頭，而不是項目符號。

在原始 markdown 中，列舉編號是阿拉伯數字後面接著一個句點與空白。數字本身代表的數值會被忽略，因此下面兩個清單並無差別：

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

上下兩個清單的輸出是相同的。

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

#### Extension: `fancy_lists`

與原始 markdown 不同的是，Pandoc 除了使用阿拉伯數字作為有序清單的編號外，也可以使用大寫或小寫的英文字母，以及羅馬數字。清單標記可以用括號包住，也可以單獨一個右括號，抑或是句號。如果清單標記是大寫字母接著一個句號，句號後請使用至少兩個空白字元。[1]

[1]之所以有這條規則，主要是要避免以人名頭文字縮寫作為開頭的段落所帶來的混淆，像是

```
B. Russell was an English philosopher.
```

這樣就不會被當作清單項目了。
這條規則並不會避免以下

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

這樣的敘述被解釋成清單項目。在這情形下，可以使用反斜線：

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

#### Extension: **startnum**

除了清單標記外，Pandoc 也能判讀清單的起始編號，這兩項資訊都會保留於輸出格式中。舉例來說，下面的輸入可以產生一個從編號 9 開始，以單括號為編號標記的清單，底下還跟著一個小寫羅馬數字的子清單：

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

當遇到不同形式的清單標記時，Pandoc 會重新開始一個新的清單。所以，以下的輸入會產生三份清單：

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

如果需要預設的有序清單標記符號，可以使用 `#.`：

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

### 定義清單

#### Extension: **definition_lists**

Pandoc 支援定義清單，其語法的靈感來自於 PHP Markdown Extra 以及 reStructuredText：[2]

```
Term 1

:   Definition 1

Term 2 with *inline markup*

:   Definition 2

        { some code, part of Definition 2 }

    Third paragraph of definition 2.
```

每個專有名詞 (term) 都必須單獨存在於一行，後面可以接著一個空白行，也可以省略，但一定要接上一或多筆定義內容。一筆定義需由一個冒號或波浪線作開頭，可以接上一或兩個空白作為縮進。定義本身的內容主體（包括接在冒號或波浪線後的第一行）應該以四個空白縮進。一個專有名詞可以有多個定義，而每個定義可以包含一或多個區塊元素（段落、代碼區塊、清單等），每個區塊元素都要縮進四個空白或一個 tab。

[2] David Wheeler 對於 markdown 的建議也同時影響了我。

如果你在定義內容後面留下空白行（如同上面的範例），那麼該段定義會被當作段落處理。在某些輸出格式中，這意謂著成對的專有名詞與定義內容間會有較大的空白間距。在定義與定義之間，以及定義與下個專有名詞間不要留空白行，即可產生一個比較緊湊的定義清單：

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

### 編號範例清單

#### Extension: **example_lists**

這個特別的清單標記 `@` 可以用來產生連續編號的範例清單。清單中第一個以 `@` 標記的項目會被編號為 '1'，接著編號為 '2'，依此類推，直到文件結束。範例項目的編號不會侷限於單一清單中，而是文件中所有以 `@` 為標記的項目均會次序遞增其編號，直到最後一個。舉例如下：

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

Explanation of examples.

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

編號範例可以加上標籤，並且在文件的其他地方作參照：

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

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

標籤可以是由任何英文字母、底線或是連字符號所組成的字串。

### 緊湊與寬鬆清單

在與清單相關的「邊界處理」上，Pandoc 與 `Markdown.pl` 有著不同的處理結果。考慮如下代碼：

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

+   Third
```

Pandoc 會將以上清單轉換為「緊湊清單」（在 "First", "Second" 或 "Third" 之中沒有 `<p>` 標籤），而 markdown 則會在 "Second" 與 "Third" （但不包含 "First"）裡面置入 `<p>` 標籤，這是因為 "Third" 之前的空白行而造成的結果。Pandoc 依循著一個簡單規則：如果文字後面跟著空白行，那麼就會被視為段落。既然 "Second" 後面是跟

著一個清單，而非空白行，那麼就不會被視為段落了。至於子清單的後面是不是跟著空白行，那就無關緊要了。(注意：即使是設定為 `markdown_strict` 格式，Pandoc 仍是依以上方式處理清單項目是否為段落的判定。這個處理方式與 markdown 官方語法規範裡的描述一致，然而卻與 `Markdown.pl` 的處理不同。)

#### 結束一個清單

如果你在清單之後放入一個縮排的代碼區塊，會有什麼結果?

```
-   item one
-   item two

    { my code block }
```

問題大了！這邊 pandoc（其他的 markdown 實作也是如此）會將 `{ my code block }` 視為 `item two` 這個清單項目的第二個段落來處理，而不會將其視為一個代碼區塊。

要在 `item two` 之後「切斷」清單，你可以插入一些沒有縮排、輸出時也不可見的內容，例如 HTML 的註解：

```
-   item one
-   item two

<!-- end of list -->

    { my code block }
```

當你想要兩個各自獨立的清單，而非一個大且連續的清單時，也可以運用同樣的技巧：

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

<!-- -->

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

### 分隔線

一行中若包含三個以上的 `*`, `-` 或 `_` 符號（中間可以以空白字元分隔），則會產生一條分隔線：

```
*  *  *  *

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

## 表格

有四種表格的形式可以使用。前三種適用於等寬字型的編輯環境，例如 Courier。第四種則不需要直行的對齊，因此可以在比例字型的環境下使用。

### 簡單表格

Extension: **simple_tables**, **table_captions**

簡單表格看起來像這樣子：

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

Table:  Demonstration of simple table syntax.
```

表頭與資料列分別以一行為單位。直行的對齊則依照表頭的文字和其底下虛線的相對位置來決定：[3]

- 如果虛線與表頭文字的右側有切齊，而左側比表頭文字還長，則該直行為靠右對齊。
- 如果虛線與表頭文字的左側有切齊，而右側比表頭文字還長，則該直行為靠左對齊。
- 如果虛線的兩側都比表頭文字長，則該直行為置中對齊。
- 如果虛線與表頭文字的兩側都有切齊，則會套用預設的對齊方式（在大多數情況下，這將會是靠左對齊）。

表格底下必須接著一個空白行，或是一行虛線後再一個空白行。表格標題為可選的（上面的範例中有出現）。標題需是一個以 `Table:` （或單純只有 `:`）開頭作為前綴的段落，輸出時前綴的這部份會被去除掉。表格標題可以放在表格之前或之後。

表頭也可以省略，在省略表頭的情況下，表格下方必須加上一行虛線以清楚標明表格的範圍。例如：

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

當省略表頭時，直行的對齊會以表格內容的第一行資料列決定。所以，以上面的表格為例，各直行的對齊依序會是靠右、靠左、置中以及靠右對齊。

### 多行表格

Extension: **multiline_tables**, **table_captions**

[3]這個方案是由 Michel Fortin 在 Markdown discussion list 的討論中所提出。

多行表格允許表頭與表格資料格的文字能以複數行呈現（但不支援橫跨多欄或縱跨多列的資料格）。以下為範例：

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

看起來很像簡單表格，但兩者間有以下差別：

- 在表頭文字之前，必須以一列虛線作為開頭（除非有省略表頭）。
- 必須以一列虛線作為表格結尾，之後接一個空白行。
- 資料列與資料列之間以空白行隔開。

在多行表格中，表格分析器會計算各直行的欄寬，並在輸出時盡可能維持各直行在原始文件中的相對比例。因此，要是你覺得某些欄位在輸出時不夠寬，你可以在 markdown 的原始檔中加寬一點。

和簡單表格一樣，表頭在多行表格中也是可以省略的：

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

多行表格中可以單只包含一個資料列，但該資料列之後必須接著一個空白行（然後才是標示表格結尾的一行虛線）。如果沒有此空白行，此表格將會被解讀成簡單表格。

#### 格框表格

Extension: **grid_tables**, **table_captions**

格框表格看起來像這樣：

```
: Sample grid table.

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

以 = 串成的一行區分了表頭與表格本體，這在沒有表頭的表格中也是可以省略的。在格框表格中的資料格可以包含任意的區塊元素（複數段落、代碼區塊、清單等等）。不支援對齊，也不支援橫跨多欄或縱跨多列的資料格。格框表格可以在 Emacs table mode 下輕鬆建立。

### 管線表格

#### Extension: **pipe_tables**, **table_captions**

管線表格看起來像這樣：

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

  : Demonstration of simple table syntax.
```

這個語法與 PHP markdown extra 中的表格語法 相同。開始與結尾的管線字元是可選的，但各直行間則必須以管線區隔。上面範例中的冒號表明了對齊方式。表頭可以省略，但表頭下的水平虛線必須保留，因為虛線上定義了資料欄的對齊方式。

因為管線界定了各欄之間的邊界，表格的原始碼並不需要像上面例子中各欄之間保持直行對齊。所以，底下一樣是個完全合法（雖然醜陋）的管線表格：

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

管線表格的資料格不能包含如段落、清單之類的區塊元素，也不能包含複數行文字。

注意：Pandoc 也可以看得懂以下形式的管線表格，這是由 Emacs 的 orgtbl-mod 所繪製：

```
| One | Two   |
|-----+-------|
```

```
| my   | table |
| is   | nice  |
```

主要的差別在於以 + 取代了部分的 |。其他的 orgtbl 功能並未支援。如果要指定非預設的直行對齊形式，你仍然需要在上面的表格中自行加入冒號。

## 文件標題區塊

（譯註：本節中提到的「標題」均指 Title，而非 Headers）

Extension: **pandoc_title_block**

如果檔案以文件標題（Title）區塊開頭

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

這部份將不會作為一般文字處理，而會以書目資訊的方式解析。(這可用在像是單一 LaTeX 或是 HTML 輸出文件的書名上。）這個區塊僅能包含標題，或是標題與作者，或是標題、作者與日期。如果你只想包含作者卻不想包含標題，或是只有標題與日期而沒有作者，你得利用空白行：

```
%
% Author

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

標題可以包含多行文字，但接續行必須以空白字元開頭，像是：

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

如果文件有多個作者，作者也可以分列在不同行並以空白字元作開頭，或是以分號間隔，或是兩者並行。所以，下列各種寫法得到的結果都是相同的：

```
% Author One
  Author Two

% Author One; Author Two

% Author One;
  Author Two
```

日期就只能寫在一行之內。

所有這三個 metadata 欄位都可以包含標準的行內格式（斜體、連結、腳註等等）。

文件標題區塊一定會被分析處理，但只有在 `--standaline`(`-s`) 選項被設定時才會影響輸出內容。在輸出 HTML 時，文件標題會出現的地方有兩個：一個是在文件的 `<head>` 區塊裡－－這會顯示在瀏覽器的視窗標

題上－－另外一個是文件的 <body> 區塊最前面。位於 <head> 裡的文件標題可以選擇性地加上前綴文字(透過 --title-prefix 或 -T 選項)。而在 <body> 裡的文件標題會以 H1 元素呈現，並附帶 “title” 類別(class)，這樣就能藉由 CSS 來隱藏顯示或重新定義格式。如果以 -T 選項指定了標題前綴文字，卻沒有設定文件標題區塊裡的標題，那麼前綴文字本身就會被當作是 HTML 的文件標題。

而 man page 的輸出器會分析文件標題區塊的標題行，以解出標題、man page section number，以及其他頁眉(header) 頁腳 (footer) 所需要的資訊。一般會假設標題行的第一個單字為標題，標題後也許會緊接著一個以括號包住的單一數字，代表 section number（標題與括號之間沒有空白)。在此之後的其他文字則為頁腳與頁眉文字。頁腳與頁眉文字之間是以單獨的一個管線符號 (|) 作為區隔。所以，

```
% PANDOC(1)
```

將會產生一份標題為 PANDOC 且 section 為 1 的 man page。

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

產生的 man page 會再加上 “Pandoc User Manuals” 在頁腳處。

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

產生的 man page 會再加上 “Version 4.0” 在頁眉處。

## 反斜線跳脫字元

Extension: **all_symbols_escapable**

除了在代碼區塊或行內代碼之外，任何標點符號或空白字元前面只要加上一個反斜線，都能使其保留字面原義，而不會進行格式的轉義解讀。因此，舉例來說，下面的寫法

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

輸出後會得到

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

而不是

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

這條規則比原始的 markdown 規則來得好記許多，原始規則中，只有以下字元才有支援反斜線跳脫，不作進一步轉義：

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

(然而，如果使用了 markdown_strict 格式，那麼就會採用原始的 markdown 規則)

一個反斜線之後的空白字元會被解釋為不斷行的空白 (nonbreaking space)。這在 TeX 的輸出中會顯示為 ~，而在 HTML 與 XML 則是顯示為 \  或 \ 。

一個反斜線之後的換行字元（例如反斜線符號出現在一行的最尾端）則會被解釋為強制換行。這在 TeX 的輸出中會顯示為 \\，而在 HTML 裡則是 <br />。相對於原始 markdown 是以在行尾加上兩個空白字元這種「看不見」的方式進行強制換行，反斜線接換行字元會是比較好的替代方案。

反斜線跳脫字元在代碼上下文中不起任何作用。

## 智慧型標點符號

**Extension**

如果指定了 `--smart` 選項，pandoc 將會輸出正式印刷用的標點符號，像是將 straight quotes 轉換為 curly quotes[4]、`---` 轉為破折號 (em–dashes)，`--` 轉為連接號 (en–dashes)，以及將 `...` 轉為刪節號。不斷行空格 (Nonbreaking spaces) 將會插入某些縮寫詞之後，例如 “Mr.”。

注意：如果你的 LaTeX template 使用了 `csquotes` 套件，pandoc 會自動偵測並且使用 `\enquote{...}` 在引言文字上。

## 行內格式

### 強調

要 強調某些文字，只要以 `*` 或 `_` 符號前後包住即可，像這樣：

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

重複兩個 `*` 或 `_` 符號以產生 **更強烈的強調**：

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

一個前後以空白字元包住，或是前面加上反斜線的 `*` 或 `_` 符號，都不會轉換為強調格式：

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

**Extension: `intraword_underscores`**

因為 `_` 字元有時會使用在單字或是 ID 之中，所以 pandoc 不會把被字母包住的 `_` 解讀為強調標記。如果有需要特別強調單字中的一部分，就用 `*`：

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

### 刪除線

**Extension: `strikeout`**

要將一段文字加上水平線作為刪除效果，將該段文字前後以 `~~` 包住即可。例如，

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

### 上標與下標

**Extension: `superscript`, `subscript`**

[4]譯註：straight quotes 指的是左右兩側都長得一樣的引號，例如我們直接在鍵盤上打出來的單引號或雙引號；curly quotes 則是左右兩側不同，有從兩側向內包夾視覺效果的引號。

要輸入上標可以用 ^ 字元將要上標的文字包起來；要輸入下標可以用 ~ 字元將要下標的文字包起來。直接看範例，

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

如果要上標或下標的文字中包含了空白，那麼這個空白字元之前必須加上反斜線。（這是為了避免一般使用下的 ~ 和 ^ 在非預期的情況下產生出意外的上標或下標。）所以，如果你想要讓字母 P 後面跟著下標文字 'a cat'，那麼就要輸入 `P~a\ cat~`，而不是 `P~a cat~`。

### 字面文字

要讓一小段文字直接以其字面形式呈現，可以用反引號將其包住：

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

如果字面文字中也包含了反引號，那就使用雙重反引號包住：

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

(在起始反引號後的空白以及結束反引號前的空白都會被忽略。)

一般性的規則如下，字面文字區段是以連續的反引號字元作為開始（反引號後的空白字元為可選），一直到同樣數目的反引號字元出現才結束（反引號前的空白字元也為可選）。

要注意的是，反斜線跳脫字元（以及其他 markdown 結構）在字面文字的上下文中是沒有效果的：

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

Extension: **inline_code_attributes**

與圍欄代碼區塊一樣，字面文字也可以附加屬性：

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

## 數學

Extension: **tex_math_dollars**

所有介於兩個 `$` 字元之間的內容將會被視為 TeX 數學公式處理。開頭的 `$` 右側必須立刻接上任意文字，而結尾 `$` 的左側同樣也必須緊挨著文字。這樣一來，`$20,000 and $30,000` 就不會被當作數學公式處理了。如果基於某些原因，有必須使用 `$` 符號將其他文字括住的需求時，那麼可以在 `$` 前使用反斜線跳脫字元，這樣 `$` 就不會被當作數學公式的分隔符。

TeX 數學公式會在所有輸出格式中印出。至於會以什麼方式演算編排 (render) 則取決於輸出的格式：

**Markdown, LaTeX, Org-Mode, ConTeXt** 公式會以字面文字呈現在兩個 `$` 符號之間。

**reStructuredText** 公式會使用 此處 所描述的 `:math:` 這個 "interpreted text role" 來進行演算編排。

**AsciiDoc** 公式會以 `latexmath:[...]` 演算編排。

**Texinfo** 公式會在 `@math` 指令中演算編排。

**groff man** 公式會以去掉 `$` 後的字面文字演算編排。

**MediaWiki** 公式會在 `<math>` 標籤中演算編排。

**Textile** 公式會在 `<span class="math">` 標籤中演算編排。

**RTF, OpenDocument, ODT** 如果可以的話，公式會以 unicode 字元演算編排，不然就直接使用字面字元。

**Docbook** 如果使用了 `--mathml` 旗標，公式就會在 `inlineequation` 或 `informalequation` 標籤中使用 mathml 演算編排。否則就會盡可能使用 unicode 字元演算編排。

**Docx** 公式會以 OMML 數學標記的方式演算編排。

**FictionBook2** 如果有使用 `--webtex` 選項，公式會以 Google Charts 或其他相容的網路服務演算編排為圖片，並下載嵌入於電子書中。否則就會以字面文字顯示。

**HTML, Slidy, DZSlides, S5, EPUB** 公式會依照以下命令列選項的設置，以不同的方法演算編排為 HTML 代碼。

1. 預設方式是將 TeX 數學公式盡可能地以 unicode 字元演算編排，如同 RTF、DocBook 以及 OpenDocument 的輸出。公式會被放在附有屬性 `class="math"` 的 `span` 標籤內，所以可以在需要時給予不同的樣式，使其突出於周遭的文字內容。

2. 如果使用了 `--latexmathml` 選項，TeX 數學公式會被顯示於 `$` 或 `$$` 字元中，並放在附帶 `LaTeX` 類別的 `<span>` 標籤裡。這段內容會用 LaTeXMathML script 演算編排為數學公式。（這個方法無法適用於所有瀏覽器，但在 Firefox 中是有效的。在不支援 LaTeXMathML 的瀏覽器中，TeX 數學公式會單純的以兩個 `$` 字元間的字面文字呈現。）

3. 如果使用了 `--jsmath` 選項，TeX 數學公式會放在 `<span>` 標籤（用於行內數學公式）或 `<div>` 標籤（用於區塊數學公式）中，並附帶類別屬性 `math`。這段內容會使用 jsMath script 來演算編排。

4. 如果使用了 `--mimetex` 選項，mimeTeX CGI script 會被呼叫來產生每個 TeX 數學公式的圖片。這適用於所有瀏覽器。`--mimetex` 選項有一個可選的 URL 參數。如果沒有指定 URL，它會假設 mimeTeX CGI script 的位置在 `/cgi-bin/mimetex.cig`。

5. 如果使用了 `--gladtex` 選項，TeX 數學公式在 HTML 的輸出中會被 `<eq>` 標籤包住。產生的 `htex` 檔案之後可以透過 gladTeX 處理，這會針對每個數學公式生成圖片，並於最後生成一個包含這些圖片連結的 `html` 檔案。所以，整個處理流程如下：

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

6. 如果使用了 `--webtex` 選項，TeX 數學公式會被轉換為 `<img>` 標籤並連結到一個用以轉換公式為圖片的外部 script。公式將會編碼為 URL 可接受格式並且與指定的 URL 參數串接。如果沒有指定 URL，那麼將會使用 Google Chart API (`http://chart.apis.google.com/chart?cht=tx&chl=`)。

7. 如果使用了 `--mathjax` 選項，TeX 數學公式將會被包在 `\(...\)`（用於行內數學公式）或 `\[...\]`（用於區塊數學公式）之間顯示，並且放在附帶類別 `math` 的 `<span>` 標籤之中。這段內容會使用 MathJax script 演算編排為頁面上的數學公式。

### Raw HTML

#### Extension: **raw_html**

Markdown 允許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處，此時的 <, > 和 & 都會按其字面意義顯示）。(技術上而言這不算擴充功能，因為原始 markdown 本身就有提供此功能，但做成擴充形式便可以在有特殊需要的時候關閉此功能。)

輸出 HTML, S5, Slidy, Slideous, DZSlides, EPUB, Markdown 以及 Textile 等格式時，原始 HTML 代碼會不作修改地保留至輸出檔案中；而其他格式的輸出內容則會將原始 HTML 代碼去除掉。

#### Extension: **markdown_in_html_blocks**

原始 markdown 允許你插入 HTML「區塊」：所謂的 HTML 區塊是指，上下各由一個空白行所隔開，開始與結尾均由所在行最左側開始的一連串對稱均衡的 HTML 標籤。在這個區塊中，任何內容都會當作是 HTML 來分析，而不再視為 markdown；所以（舉例來說），* 符號就不再代表強調。

當指定格式為 markdown_strict 時，Pandoc 會以上述方式處理；但預設情況下，Pandoc 能夠以 markdown 語法解讀 HTML 區塊標籤中的內容。舉例說明，Pandoc 能夠將底下這段

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

轉換為

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

而 Markdown.pl 則是保留該段原樣。

這個規則只有一個例外：那就是介於 <script> 與 <style> 之間的文字都不會被拿來當作 markdown 解讀。

這邊與原始 markdown 的分歧，主要是為了讓 markdown 能夠更便利地混入 HTML 區塊元素。比方說，一段 markdown 文字可以用  標籤將其前後包住來進行樣式指定，而不用擔心裡面的 markdown 不會被解譯到。

### 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 代碼。行內 TeX 指令會被保留並不作修改地輸出至 LaTeX 與 ConTeXt 格式中。所以，舉例來說，你可以使用 LaTeX 來導入 BibTeX 的引用文獻：

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

請注意在 LaTeX 環境下時，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` 與 `end` 標籤之間的內容，都會被當作是原始 LaTeX 資料解讀，而不會視為 markdown。

行內 LaTeX 在輸出至 Markdown, LaTeX 及 ConTeXt 之外的格式時會被忽略掉。

## LaTeX 巨集

Extension: **`latex_macros`**

當輸出格式不是 LaTeX 時，pandoc 會分析 LaTeX 的 `\newcommand` 和 `\renewcommand` 定義，並套用其產生的巨集到所有 LaTeX 數學公式中。所以，舉例來說，下列指令對於所有的輸出格式均有作用，而非僅僅作用於 LaTeX 格式：

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

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

在 LaTeX 的輸出中，`\newcommand` 定義會單純不作修改地保留至輸出結果。

## 連結

Markdown 接受以下數種指定連結的方式。

### 自動連結

如果你用角括號將一段 URL 或是 email 位址包起來，它會自動轉換成連結：

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

### 行內連結

一個行內連結包含了位在方括號中的連結文字，以及方括號後以圓括號包起來的 URL。（你可以選擇性地在 URL 後面加入連結標題，標題文字要放在引號之中。）

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

方括號與圓括號之間不能有空白。連結文字可以包含格式（例如強調），但連結標題則否。

### 參考連結

一個 **明確**的參考連結包含兩個部分，連結本身以及連結定義，其中連結定義可以放在文件的任何地方（不論是放在連結所在處之前或之後）。

連結本身是由兩組方括號所組成，第一組方括號中為連結文字，第二組為連結標籤。（在兩個方括號間可以有空白。）連結定義則是以方括號框住的連結標籤作開頭，後面跟著一個冒號一個空白，再接著一個 URL，最後可以選擇性地（在一個空白之後）加入由引號或是圓括號包住的連結標題。

以下是一些範例：

```
[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'
```

連結的 URL 也可以選擇性地以角括號包住：

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

連結標題可以放在第二行：

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

需注意連結標籤並不區分大小寫。所以下面的例子會建立合法的連結：

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

[Foo]: /bar/baz
```

在一個 **隱性**參考連結中，第二組方括號的內容是空的，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` 以及大多數其他 markdown 實作中，參考連結的定義不能存在於嵌套結構中，例如清單項目或是區塊引言。Pandoc lifts this arbitrary seeming restriction。所以雖然下面的語法在幾乎所有其他實作中都是錯誤的，但在 pandoc 中可以正確處理：

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

### 內部連結

要連結到同一份文件的其他章節，可使用自動產生的 ID（參見HTML, LaTeX 與 ConTeXt 的標題識別符 一節後半）。例如：

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

或是

```
See the [Introduction].

[Introduction]: #introduction
```

內部連結目前支援的格式有 HTML（包括 HTML slide shows 與 EPUB）、LaTeX 以及 ConTeXt。

## 圖片

在連結語法的前面加上一個 ! 就是圖片的語法了。連結文字將會作為圖片的替代文字（alt text）：

```
![la lune](lalune.jpg "Voyage to the moon")

![movie reel]

[movie reel]: movie.gif
```

### 附上說明的圖片

#### Extension: **implicit_figures**

一個圖片若自身單獨存在一個段落中，那麼將會以附上圖片說明 (caption) 的圖表 (figure) 形式呈現。[5]（在 LaTeX 中，會使用圖表環境；在 HTML 中，圖片會被放在具有 `figure` 類別的 `div` 元素中，並會附上一個具有 `caption` 類別的 `p` 元素。）圖片的替代文字同時也會用來作為圖片說明。

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

如果你只是想要個一般的行內圖片，那麼只要讓圖片不是段落裡唯一的元素即可。一個簡單的方法是在圖片後面插入一個不斷行空格：

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

## 腳註

#### Extension: **footnotes**

Pandoc's markdown 支援腳註功能，使用以下的語法：

[5]這項功能尚未在 RTF, OpenDocument 或 ODT 格式上實現。在這些格式中，你會得到一個在段落中只包含自己的圖片，而無圖片說明。

```
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.
```

腳註參考用的 ID 不得包含空白、tabs 或換行字元。這些 ID 只會用來建立腳註位置與腳註文字的對應關連；在輸出時，腳註將會依序遞增編號。

腳註本身不需要放在文件的最後面。它們可以放在文件裡的任何地方，但不能被放入區塊元素（清單、區塊引言、表格等）之中。

#### Extension: `inline_notes`

Pandoc 也支援了行內腳註（儘管，與一般腳註不同，行內腳註不能包含多個段落）。其語法如下：

```
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.]
```

行內與一般腳註可以自由交錯使用。

## 引用

#### Extension: `citations`

Pandoc 能夠以數種形式自動產生引用與參考書目（使用 Andrea Rossato 的 `hs-citeproc`）。為了使用這項功能，你需要一個下列其中一種格式的參考書目資料庫：

| Format | File extension |
|---|---|
| MODS | .mods |
| BibLaTeX | .bib |
| BibTeX | .bibtex |
| RIS | .ris |
| EndNote | .enl |
| EndNote XML | .xml |

| Format | File extension |
|---|---|
| ISI | .wos |
| MEDLINE | .medline |
| Copac | .copac |
| JSON citeproc | .json |

需注意的是副檔名 `.bib` 一般而言同時適用於 BibTeX 與 BibLaTeX 的檔案，不過你可以使用 `.bibtex` 來強制指定 BibTeX。

你需要使用命令列選項 `--bibliography` 來指定參考書目檔案（如果有多個書目檔就得反覆指定）。

預設情況下，pandoc 會在引用文獻與參考書目中使用芝加哥「作者－日期」格式。要使用其他的格式，你需要用 `--csl` 選項來指定一個 CSL 1.0 格式的檔案。關於建立與修改 CSL 格式的入門可以在 http://citationstyles.org/downloads/primer.html 這邊找到。https://github.com/citation-style-language/styles 是 CSL 格式的檔案庫。也可以在 http://zotero.org/styles 以簡單的方式瀏覽。

引用資訊放在方括號中，以分號區隔。每一條引用都會有個 key，由 `@` 加上資料庫中的引用 ID 組成，並且可以選擇性地包含前綴、定位以及後綴。以下是一些範例：

```
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 `@` 前面的減號 (`-`) 將會避免作者名字在引用中出現。這可以用在已經提及作者的文章場合中：

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

你也可以在文字中直接插入引用資訊，方式如下：

```
@smith04 says blah.

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

如果引用格式檔需要產生一份引用作品的清單，這份清單會被放在文件的最後面。一般而言，你需要以一個適當的標題結束你的文件：

```
last paragraph...

# References
```

如此一來參考書目就會被放在這個標題後面了。