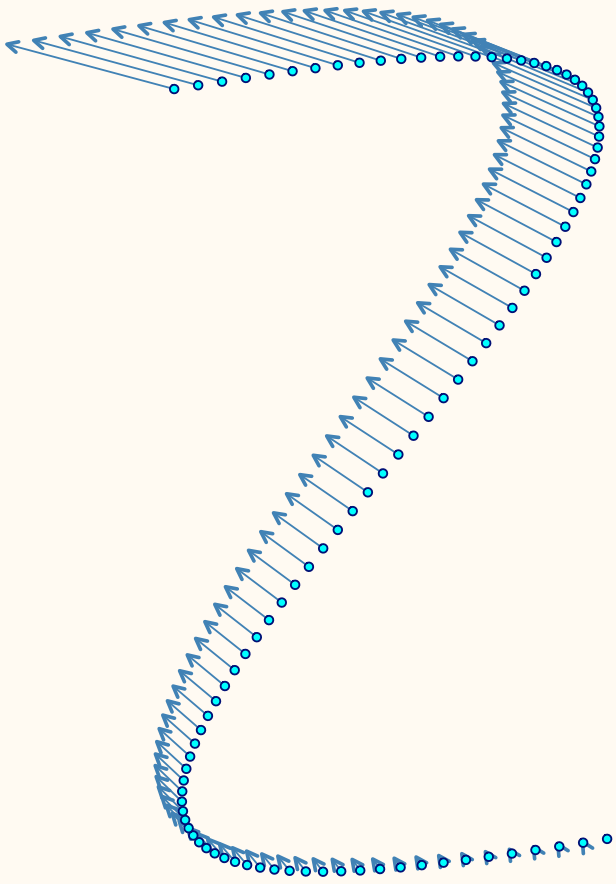
大家來學 LaTeX

Version 1.0

2020 年 12 月 20 日



By Edward G.J. Lee 李果正

Email : edt1023@info.sayya.org

本教學文件之製作，部份接受下列計劃補助：

行政院研考會委辦「政府機關資料文件交換之電子檔案格式應用研究」

目 錄

第 1 章

先來說一些故事

TEX 是 Donald E. Knuth[1] 教授的精心傑作，它是個功能非常強大的幕後排版系統，含有彈性很大，而且很低階的排版語言。當初，是因爲 Knuth 教授在寫他的大著 TAOCP(*The Art of Computer Programming*) 時，發覺書商把他書中的數學式子排得太難看了，於是決定自行開發一個非常適合排數學式子的排版語言，這就是 TEX 系統的來由。

不僅僅是談到 TEX 一定會提到 Knuth 教授，只要提到排版，沒有人可以忽略他的 TEX 所帶來的變革、影響，甚至 TEX 已經 20 幾歲了，仍然深深影響著排版界及排版軟體，可見這個排版軟體眞的是非同小可。

1.1 Knuth 教授的生平簡介

1938.01.10	誕生。Milwaukee, Wisconsin; U.S. citizen。
1956	進入凱思工學院（Case Institute of Technology）學習物理。
1960	畢業後進入加州理工學院（California Institute of Technology）研究所，此時轉向數學領域的研究方向。
1961.06.24	和 Nancy Jill Carter 結婚。他的中文名字是高德納，他老婆名叫高精蘭，老婆小他一歲。兩個小孩，一男一女。中文名字是 1977 去中國大陸時取的。
1963	拿到 Ph.D.，並留校任教。
1968	開始任教於 Stanford 大學，資訊科學系 (Computer Science)。同年開始撰寫名聞遐邇的 TAOCP(*The Art of Computer Programming*)。有人曾說，看了這部書，往後對寫程式的話題都會變得謙虛。:)
1977	不滿意書商所印出的 TAOCP，因此，自行開發 TEX 排版系統，這可就影響了往後的排版、出版界，至今不墜。但也因此拖延了 TAOCP 第四冊的完成時間。
1986	榮獲 ACM 軟體系統獎。

他可說是著作等身，書籍、論文都有，他的任何著作有個奇怪的『副作用』，那就是任何

[1] http://www-cs-faculty.stanford.edu/~knuth/

人發現書上的錯誤，都可以向他舉發，並領取 $2.56（美金）！想試試看「手氣」嗎？台灣就有人領過。:-) 爲什麼是 $2.56？Knuth 教授的答案是：

> *"256 pennies is one hexadecimal dollar."*

發現 TEX 系統的臭蟲也是一樣，這個獎金每年倍數增加，直至 $655.36($2^{16}$ pennies) 爲止。

他也很推崇自由軟體基金會 (Free Software Foundation)[2] 及 GNU/Linux，把一些希望都寄託於 GNU/Linux，尤其是 Unicode 環境，他希望 GNU/Linux 很快就能在網頁上顯示他的中文名字，而不必使用圖檔。其實是可以做到了，只是 Unicode 環境還不算普及罷了。他曾在 1996 年接受 Dr. Dobb's Journal 訪問時英雄惜英雄的公開表示，創導自由軟體 (Free Software)[3] 的 Richard M. Stallman 是他心目中的英雄之一，他認爲自由軟體基金會這些人所做的貢獻很不錯，雖然和他的方式不一定一樣，但許多貢獻是互有認同的。

在發展 TEX 時就同時思考 WEB（這個詞比目前使用的 WWW Web 還早使用），那是一種 literate programming 的程式方法。他認爲目前已成熟的可以提出含有文件的程式方法，使寫程式就像寫文學作品（小說？）一般的藝術表現。後來也把他由 C 改寫（和 Silvio Levy 合作），名爲 CWEB[4]。TEX 就是由 WEB 寫成的，WEB 可視爲 Pascal 語言的一個子集。

一個 literate 程式師可被視爲文學作家、評論寫作者、隨筆作者 ……，程式的表現不僅僅是搬弄符號，而是展現自己的風格，當然也是指達成目的的風格、甚至程式中變數運用的風格。

這樣一來就可以展現讓人類較能理解的程式碼。使用形式及非形式的融合，而且兩者間相輔相成，目的達成了，也讓閱讀的人就好像讀文學作品般的去抓住作者的心，使程式創作提升至更高的（文學）藝術境界，而不再是死板板的 code 了。

Knuth 大師已於 1992 年自大學退休，但仍在 Stanford/Oxford 等大學有授課。目前正處於隱居的生活，他這麼早退休的原因，就是因爲 TAOCP 這部書，他估計大約要花 20 年來完成，因此目前的重點工作是完成他的 TAOCP（分成好幾冊，目前眞正出版的只有三大冊）。他認爲 email[5] 會影響他的思路，因此，寧願留住址，要和他聯絡就只好寫信，傳眞。給他的秘書的 email，是最後有時間才會去看，他曾公開的表明，這部書是他這一生

[2] http://www.fsf.org

[3] http://www.gnu.org/philosophy/free-sw.html

[4] http://www-cs-faculty.stanford.edu/~knuth/cweb.html

[5] http://www-cs-faculty.stanford.edu/~knuth/email.html

中最重要的工作。

雖然 Knuth 教授寫了許多嚴肅艱深的書籍、論文，但是他也是有風趣的一面，在 1996 年，*Mathematisch Centrum* (MC, 爲慶祝五十周年慶改稱爲 *Centrum voor Wiskunde en Informatica*, CWI) 曾邀請他演講，並知會荷蘭的 TeX User Group(NTG)[6]，NTG 見機會難得，就邀請 Knuth 教授另開個 TeX 及 METAFONT 討論會，並接受大家的提問，他說：『不對，我也是可以問你們問題的 !』。而且，他還說：『這種問答的內容，很可能在不同場合重複過，所以，如果我對同一個問題，曾有過兩種答案的話，你們必需取其平均值。』他的妙語如珠，在許多類似的場合常常引起哄堂大笑，但實際的內容卻絕非泛泛之言。:-)

1.1.1 TAOCP(*The Art of Computer Programming*)

這可是演算法的大著，請別折騰我，我只是心嚮往之，這部書我沒有一本是看得懂的。:-)

1. 第一冊，*Fundamental Algorithms*，1968 第一版，ISBN 0-201-89683-4。
2. 第二冊，*Seminumerical Algorithms*，1969 第一版，ISBN 0-201-89684-2。
3. 第三冊，*Sorting and Searching*，1973 第一版，ISBN 0-201-89685-0。
4. 第四冊，*Combinatorial Algorithms*，尚未完成，可能會先出分冊。
   (a) 分冊 4A, *Enumeration and Backtracking*
   (b) 分冊 4B, *Graph and Network Algorithms*
   (c) 分冊 4C, *Optimization and Recursion*
5. 第五冊，*Syntactic Algorithms*，計劃 2010 年完成。
6. 第六冊，the theory of context-free languages，書名可能會變更，尚未開始。
7. 第七冊，Compiler techniques，書名可能會變更，尚未開始。

詳細的目錄大綱及修訂版的資訊，請參考網頁上的資料：

http://www-cs-staff.stanford.edu/~knuth/taocp.html

1.2 那麼，LaTeX 又是什麼呢？

TeX 是個很低階的排版語言，如果排版時都要從這種低階語言來控制版面的話，那將會非常的煩複，所以，一些經常要用到的功能，都會先去定義好（稱爲巨集，macro），這樣排版時才會方便、快速，直接引用已定義好的巨集裡頭的指令就可以了。

[6] http://www.ntg.nl，荷蘭的 TeX User Group 算是相當活躍的。

原始的 TeX 已經有了一組 macro，是 Knuth 教授所寫的，那就是著名的 Plain TeX，但仍然不夠方便、直觀，於是 Leslie Lamport[7] 又寫了另一組的 macro，稱爲 LaTeX，主要是把版面配置和文章內容，適度的分開處理，只要使用者選定了一種類別，整本書或整篇文章的結構就是按照這個類別來安排版面，這樣寫文件的人只要專注於文章內容就可以了，版面配置就完全交給 TeX/LaTeX 去處理。

既然 LaTeX 只不過是 TeX 的一大組巨集，那，當然原來的 TeX 的指令，大部份也是可以用在 LaTeX 文稿當中的。而且，LaTeX 並不是目前唯一的 TeX macro，其他如 eplain TeX, ConTeXt, TeXinfo 等都是 TeX macro，也都有他們自成一套的語法。目前的 LaTeX 由 LaTeX 3 Project[8] 所維護及發展。

如果談到這裡，你還是霧煞煞的話，類比成 HTML markup 標記語言就能大概知道一些概念了，當然，這和 HTML 標記語言是可相提而無法並論的。如果連 HTML 也不熟悉，那也沒關係啦！這章本來就是在說故事嘛！:-) 只要繼續看後面的內容就行了。

1.3 一般人對 LaTeX 的迷思

這裡引用 Peter Flynn 在他的 *A beginner's introduction to typesetting with LaTeX*[9] 一文中所提出來的六大迷思，並添加個人的一些看法：

- LaTeX 只能使用一種字型？
  當然不是，雖然 TeX 系統預設是使用當初 Knuth 教授所設計的 METAFONT，但目前 OpenType, TrueType, Adobe Type1 字型都可以用在 LaTeX 當中，只不過，安裝字型的部份不是那麼的直觀就是了，但比起其他的排版系統，TeX/LaTeX 所能利用的字型種類，可以說是最多的。

- LaTeX 只能用於 Unix-like 的作業系統？
  Knuth 教授慷慨的把 TeX 的原始程式碼開放出來[10]，所以，只要是有人在使用的作業系統都可以移植過去，不必擔心版權的問題。像 MS DOS, OS/2, MS Windows,

[7] Leslie Lamport 於 1985 在 CSL 任職時寫了 LaTeX。目前任職於 DEC Systems Research Center 但已幾乎沒有參與 LaTeX 的後續發展了。

[8] http://www.latex-project.org/latex3.html

[9] ftp://ftp.dante.de/tex-archive/info/beginlatex/beginlatex.pdf

[10] 雖然有一些版權的規定，但是 Knuth 教授的本意，這些原始碼是屬於 Public Domain，只是，如果經過修改，不能再以 TeX 爲名，要另外改一個名稱，以免和原來的 TeX 搞混。TeX 這個商標的所有人是美國數學協會（American Mathematical Society, AMS）。

Mac OS, Unix-like 系統都有 TeX 的移植版本，甚至是 PDA（e.g. the Sharp *Zaurus*）都有 TeX/LaTeX 的縱跡。可以說是走到哪兒，就可以用到哪兒，而且，文稿都是互通的，列印結果也相同。

- LaTeX 已經過時了？
  剛好相反，LaTeX Project[11]及其他相關 packages 正穩定的研發當中，尤其和目前新一代的 SGML/XML/HTML 及資料庫系統，都積極的想辦法銜接起來。對於數學式子的排版，至今無人能出其右。有興趣的話，可以參考 news://comp.text.tex 的流量，及其中 CTAN[12] 對於巨集更新、上傳的消息發布。

- LaTeX 沒有所見即所得（WYSIWYG）？
  某種層面上而言，是的。因爲他本質上是幕後排版系統。但是，所產生的 dvi/ps/pdf 檔，可以很精準的顯示你所想要表達的內容，不曉得這算不算是「所見即所得」？另外，一些相關 GUI 配合的軟體，也會打破幕後排版的定義，例如 LyX[13], TeXmacs[14] 等等。

- LaTeX 很難學？
  這個嘛！我只能說，有什麼東西是很好學的？如果只是一般使用，而不是當個排版專家，甚至 TeX/LaTeX programmer，那麼，幾十分鐘的說明，就可以「上工」了！剩下的只是一些細部調整的問題（不去調整，也絕對不會離譜）。相對於一般圖形化 Office 類軟體，要眞正把他的內容熟悉，恐怕也是不簡單的。另外的問題，大概是幕前、幕後系統操作習慣的問題，甚至是一種第一印象了，就好像說到電腦，有不少人的腦子裡就會浮現 MS Windows 的圖象一樣。:-)

- LaTeX 是專爲科學家或數學家而寫的？
  的確，當初 Knuth 教授是爲了表達精確、品質優美的數學式子而開發 TeX 的，但由於 TeX 的彈性，使得在其他的領域的使用者也爭相使用，已經不是局限在學術界在使用了。尤其 XML 的興起，需要一個適合的格式化工具（formatter）的配合，TeX/LaTeX 就剛好稱職的做他排版專業的工作。

[11] http://www.latex-project.org/
[12] http://tug.ctan.org/
[13] http://www.lyx.org/
[14] http://www.texmacs.org/

1.4 本文的重點方向

TEX/LATEX 的相關議題：版面的配置、排版規範、字型技術、TEX the program 的改進、繪圖技術、TEX macro 的撰寫、pre/post 處理程式的撰寫、出版流程 …… 等等，浩瀚無涯，可以說窮一輩子也可能研究不完，所以，各位在「陷入」這個領域之前，建議最好有個適當的範圍，以免「愈陷愈深」終至無法自拔。

所以，本文的重點是放在「標準」LATEX 本身，其他相關的 package/macro 除非必要，盡量不提及。但是 LATEX 本身也不是十全十美的，所以，有需要的地方也需要一並提及外來的 macro，無論如何，重點是放在標準 LATEX 本身，請閱讀本文的朋友注意一下這個方向。LATEX 本身就能解決的，就不假外求了，雖然會失去了一些彈性，但爲免造成 package/macro 滿天飛，打亂學習陣腳，剛開始也只好如此了！

而且，可能的情形下，會往一般用途的方向去介紹，而不僅僅專注在數理排版。數學式子雖是 TEX/LATEX 的拿手把戲，但不代表一般用途就不適合，相反的，現在有許多商業公司正把 TEX/LATEX 用於一般商業出版上。

第 2 章

行前準備

使用 TEX/LATEX 系統，剛開始，比較麻煩的是安裝問題。不過，以現在的作業系統而言，幾乎較流行的作業系統都有現成包好的 TEX 系統套件可以安裝，例如 Un*x 系統的 teTEX、Windows 系統的 MiKTEX 及 fpTEX。另外，也有 TEX Live CD[1] 可以供下載、購買，這是 TUG(TEX User Group)[2] 製作的各種作業系統的可執行檔，使用上相當方便。

目前所謂的 TEX 套件，是把原來的 TEX 排版引擎本身，加上 LATEX 及其他相關的巨集，再加上字型軟體（METAFONT）、繪圖程式（METAPOST）、字型檔 …… 等等，所組合成的一整個可實際運作的排版系統。因此，什麼是 TEX？會因使用的場合不同而有不同的意義，一般純指指令本身的時候，就單純寫成小寫的 `tex`，此時所用的巨集，預設就是 Knuth 教授所寫的 plain TEX。寫成 TEX 時，一般是指整個系統而言。這在 LATEX 巨集亦同，`latex` 指的是指令，LATEX 指的是整個巨集系統。

2.1 Unix-like 系統

一般 Unix-like 系統都是安裝 teTEX 套件，凡是和 `tetex` 字樣相關的 packages 都安裝起來，目前 GNU/Linux 各種 distribution 及 FreeBSD 都有現成的 packages 供安裝使用。如果是沒有提供這個套件的作業系統，可能得自行編譯了，原始碼在：

```
http://www.tug.org/teTeX
ftp://cam.ctan.org/tex-archive/systems/unix/teTeX
ftp://tug.ctan.org/tex-archive/systems/unix/teTeX
ftp://dante.ctan.org/tex-archive/systems/unix/teTeX
```

[1] http://tug.org/texlive/

[2] http://www.tug.org/。加入會員的話，只要些許會費，則可以獲得 TEX Live CD 及年報資訊的寄發。

2.2 MS Windows 系統

最常使用的 free 版本，大概就是 MiKTEX 及 fpTEX，其中，後者等於是 Un*x 中的 teTEX 的 Windows 移植版本。

http://www.miktex.org/
http://www.fptex.org/

安裝的話都自動化了，應該可以很方便的安裝起來。

2.2.1 cygwin 環境

這是 Windows 系統下的一個 Un*x 環境（正確的說，是 Linux-like），有了這個環境就可以使用 Unix-like 的界面，也可以編譯 Unix-like 中的程式，當然也就可以安裝 Unix-like 系統的 teTEX 套件了，有人習慣了 Unix-like 的操作環境，但又常需要在 Windows 平台下作業，這是個不錯的選擇。

http://sources.redhat.com/cygwin
http://sources.redhat.com/cygwin/setup.exe

只要先下載 `setup.exe` 這個可執行檔，然後執行後按著指示就可以完成安裝，當然，網路要連線。至於 teTEX 相關的套件，安裝好 cygwin 就會安裝，至於中文 CJK 套件，感謝 seventeen 的製作，請參考：

http://seventeen.mit.edu/blog/17/archives/000141.html

2.3 Mac OS X 系統

個人對 Mac OS X 並不熟悉，所以僅提供個人知道的 distribution。但 Mac OS X 亦可以安裝 Un*x 系統上的 teTEX 系統，也可以在其上自行編譯。

參考文件：
[文件]MacOS 10.2.4 安裝 teTeX
http://www.rna.nl/tex.html
http://www.cs.wright.edu/~jslater/mac-tex/mac-tex-intro/mac-tex-intro.pdf
TEXShop
http://www.uoregon.edu/~koch/texshop/texshop.html
iTEXMac
http://itexmac.sourceforge.net/

2.4 商業維護的 TEX/LATEX 系統

上面所提到的都是 free 的版本，縱使是 TEX Live CD，雖也可以讓非 TUG 會員訂購，但他們的 iso 是可以自由下載，不一定要花錢買。當然，這裡是鼓勵大家加入會員或花點小錢購買，也算是贊助他們繼續維護好品質的 TEX 系統。另外，亦有商業維護的 TEX/LATEX 系統，雖然要花錢，但功能上可能會比較符合特別的需要，而且有比較完整的售後服務方案：

TrueTEX	http://www.truetex.com/	Windows
TurboTEX	http://www.truetex.com/	Un*x, DOS, Windows
Y&Y TEX	http://www.YandY.com/	Windows
pcTEX	http://www.pctex.com/	Windows
VTEX	http://www.micropress-inc.com/	Windows, Linux, OS/2
Scientific Word	http://www.sciword.demon.co.uk	Windows
Textures	http://www.bluesky.com/	Macintosh
OzTEX	http://www.trevorrow.com/oztex/	Mac OS X
Scientific Assistant	http://www.advanced-science.com/	Mac OS X

2.5 選個順手的編輯器

TEX/LATEX 本身並不內附編輯器 ，這和許多排版軟體不一樣，他只專注在排版的過程，原始文稿是如何產生的並不插手干涉，這樣子的自由度很大，每個人都可以選用適合他自己的編輯器，但和目前一般的幕前排版系統比較的話，會讓初接觸的朋友不知所措，因爲他不曉得要如何使用 TEX/LATEX 來「編輯」原始文稿！

當然，有些 TEX/LATEX 的發行版本，乾脆就弄了個編輯器，把編輯器和排版系統本身連接起來，這樣是很方便，但有很大的可能又得花時間學習另一種不熟悉的編輯器操作。本文的說明，不準備被編輯器給綁住，你愛用什麼編輯器就用什麼編輯器，讓我們專心在排版過程本身吧！

以下僅簡單介紹和中文相容的編輯器。當然，只要使用順手的編輯器都可以拿來用。原則上，剛開始接觸 LATEX，個人是建議從命令列開始瞭解起，等整個流程有個概念後再來使用一些方便的編輯器上的巨集及按鈕設定，不然，有時編輯器上的設定有問題時，會不知道從何改起。至於編輯器的操作，請自行參考各編輯器的說明文件或網路上的教學文件，這裡就不多談了。

2.5.1 *Vim*

這有各種平台的版本可以下載：

http://www.vim.org/

可以配合 vim-latex suite 來使用：

http://vim-latex.sourceforge.net/

2.5.2 GNU Emacs/XEmacs

這也是有各種平台的版本，也可以配合 AUCTEXfootnotehttp://www.gnu.org/software/auctex/ 來使用，相當方便，這也是 Knuth 教授本身所愛用的編輯器：

http://www.gnu.org/software/emacs/emacs.html
http://www.xemacs.org

2.5.3 NEdit

這也可以配合 AUCTEXindexAUCTeX@AUCTEX 來使用，但只有 Un*x 系統的版本：

http://nedit.org

2.5.4 WinEdt

這是 shareware，只有 Windows 版本：

http://home.istar.ca/~winedt

2.5.5 UltraEdit

這也是 shareware，也是只有 Windows 版本。

http://www.ultraedit.com

2.5.6 Kile

這是很方便的 LaTeX 圖形界面整合環境，還可叫出繪圖軟體來繪圖，如果其他作業系統也有安裝 Qt/KDE 的話，也是可以編譯安裝使用：

http://kile.sourceforge.net/

第 3 章

TEX/LATEX 語法概說

這一章要談的是，和一般的純文字文稿及其他 markup 式文件系統在語言上的一般差異性。爲了讓觀念上能夠更清楚，以下所述主要是要在命令列執行的，致於編輯器上方便的按鍵及巨集，這裡就不多談，一方面是每個人使用的編輯器不一樣，二方面是要先把黑盒子拿掉，整個處理流程才會有概念。

當然，由於完全還沒有開始實際寫文稿來測試，所以，這章是紙上談兵，不必動手，用看的就好。但，別急，我們會在第 4 章開始實際玩看看，請別忘了，到時要再回頭來複習一下這些資料。

而且，前面已經說過，這篇文章主要是著眼於 LATEX 所附上的巨集，一些其他方便的套件引用，將會在最後或另文再來談。其實，不引用任何外來特殊套件，讓 LATEX 本身去處理，最起碼也就不會太離譜，要講求美觀、微調，個人是認爲先把基礎弄起來再說，有些套件的複雜程度，會令人頭疼，你是不是眞有這個需要，值得考慮。而且，很多時候自認爲不錯的「微調」，其實常常會不合排版的慣例。TEX/LATEX 的語法，可以是很簡單明白，也可以是相當的複雜，這是 TEX 系統本身的彈性所導致。

3.1 LATEX 文稿的處理流程

最簡單的一句話，就是把編輯器編輯好的文稿（通常結尾是 .tex），利用 latex 這個指令去編譯文稿就對了！

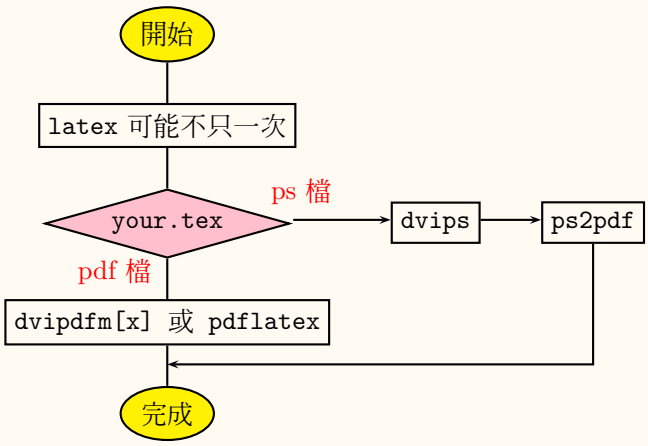
```
latex your.tex
```

需要注意的是，如果有索引，還要用 makeindex 執行一次，有參考文獻，還需要 bibtex 處理一次，最後再使用 latex 再處理一至二次，也就是說視文稿的複雜程度，latex 可能需要執行好幾次，這在往後碰到時會再提出來。另外，處理中文的話，需要其他前置處

理，這裡暫時先以英文文稿來說明，中文的部份，只要加入中文環境及（或）改用能處理中文的前置處理器就可以了。

這樣經過 latex 處理後，會產生一個 your.dvi 檔，然後可以使用 dvips 來產生 PostScript 格式的檔案。也可以使用 dvipdfm[x] 來產生 PDF 的格式，當然，也可以使用 ps2pdf 經由 PostScript 格式轉換成 PDF 格式。另外，也可由 pdflatex 由 your.tex 直接編譯成 PDF 格式的輸出。

3.1.1 總結一下處理流程



3.2 LATEX 的特殊專用符號

在 TEX/LATEX 的世界，原始文稿都是純文字檔，任何一種編輯器都可以打開來編輯、觀看。而排版指令通常是由反斜線（\, backslash）所開頭來引導。註解則是由百分號（%）來引導。例如，以編輯器編輯下列文字：

```
This is my first \LaTeX{} typesetting example.
```

編譯後會變成以下的結果：

This is my first LATEX typesetting example.

其中的 \LaTeX 就是 LATEX 的一個指令，會顯示 LATEX 這個特殊的圖示。

由於，西方國家的語系，通常字母、符號的最大容量只有 256 個（2^8），因此，許多現有的符號必須拿來當做控制指令，才能符合排版的多樣化需求。以下的符號，接觸 TEX/LATEX

的朋友，可能都得時時留意，不要未經處理就直接寫進文稿裡頭去了。

通常，編輯器的語法顏色會幫助判斷語法是否正確，但不是都能完美無缺，有時還是會漏掉，這時別忘了查看一下 *.log 檔案，例如：編譯 your.tex 檔的話，他的 log 檔就是 your.log。

符號	作用	文稿上使用	LATEX 的替代指令
\	下排版命令	$\backslash$	\textbackslash
%	註解	\%	NA
#	定義巨集	\#	NA
~	產生一個空白	\~{}	\textasciitilde
$	進入（離開）數學模式	\$	\textdollar
_	數學模式中產生下標字	_{}	\textunderscore
^	數學模式中產生上標字	\^{}	\textasciicircum
{	標示命令的作用範圍	\{	\textbraceleft
}	標示命令的作用範圍	\}	\textbraceright
<	數學模式中的小於符號	$<$	\textless
>	數學模式中的大於符號	$>$	\textgreater
|	OT1 編碼，數學模式中才能正確顯示	$\|$	\textbar
&	表格中的分隔符號	\&	NA

3.3 LATEX 排版上的一些規範或慣例

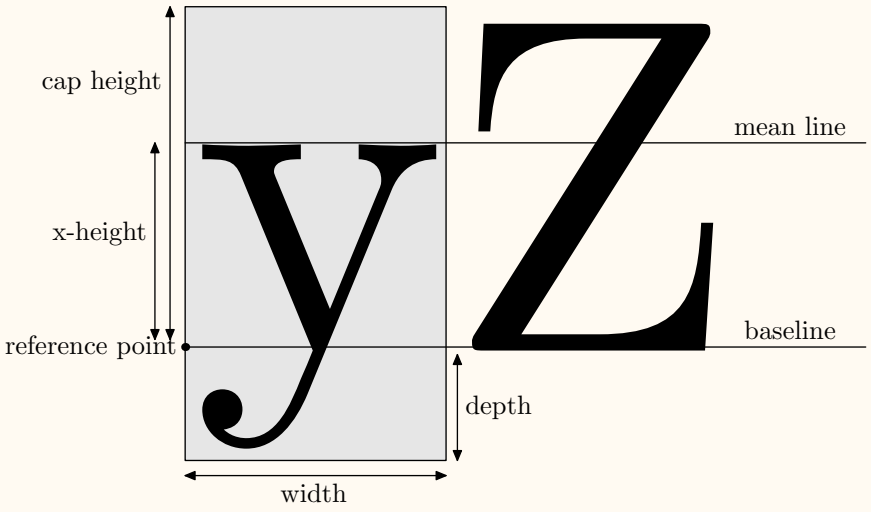
除了上面所談到的特殊符號外，也有一些規範或慣例要遵守，有些是比較硬性的規定，有些則只是慣例，可能不同的國家、語言會有不同的慣例，暫時先把他當成是 LATEX 的遊戲規則就成了。

3.3.1 字型的相關術語

要談排版上的規範、慣例前，我們得先認識一下字型的一些術語，以便往後文章中提到時有個概念。通常我們每個字都是置放於一個假想的方框中，稱爲 em-square，同一個字型的同一個點數，每一個 em-square 大小都是相同的，實際上的字（glyph）要置放在這個 em-square 的什麼位置，這是字型設計者的觀點，所以，同樣點數的不同字型，他的字的大小不一定會一樣，因爲我們是使用 em-square 的大小在比較的，而非實際的 glyph。

在文章中排列的時候，則是將 glyph 置於一個以假想參考點（reference point）爲基準的

一個假想線上，稱爲基線（baseline），大寫字母除了 Q 以外，他們的底部都是置於基線上的。但小寫字母則不一定剛好座落在基線上，有些字的筆畫可能超出基線以下，例如 y、j 等字母。關於字型在文章中的置放位置，我們來看看一個模擬圖：[1]



這個超出基線以下的長度，我們稱之爲深度（depth），以上的就稱爲字高（height），當然大小寫的不同又分爲大寫字母的字高（cap height）及小寫字母 x 的字高（x-height），由於這個例子裡是調合字，所以每個字的寬度（width）不一定會一樣，像打字機字族的則是等寬的字型。字高加上深度，我們就稱之爲 totalheight，大部份的情況，僅僅說 height 時是不包括 depth 的，而且通常指的是 cap height。

mean line 在一般比較少用到，通常是字型設計時才會用到，他是指小寫字母去除上面突出的部份所連成的一個基準線，這個 mean line 到 baseline 的距離，一般就稱爲 x-height，當然就是小寫字母 x 的高度，因此我們會有一個長度單位，稱爲 ex，指的就是這個 x-height。

中文字的話比較特殊，他是以 em-square 的中心點來置放 glyph 的，在中英文混合時，中文字並不是剛好座落於基線上的，會超出基線下一點點，至於會超出多少，則和字型的設計有關，每種字型都有可能會不同。這也是爲求排版上的一致性，字型可能都需要盡量使用同一套的各種字型的原因，否則就得經過微調，才能使整個字型表現上取得協調一致。

這些專門術語，往後提到一些指令的參數的描述時都會使用到，因此先熟悉一下，例如：字型旋轉時，跟據的就是以參考點（reference point）爲準，沿延伸出的軸心來旋轉的，而一般所說的行距，指的是上下兩 baseline 的距離。

[1]實際字型設計上的各部份專有名詞及結構，當然不會是這麼簡略，這裡的模擬圖，只是暫時讓字的一般置放有個粗略的概念。

3.3.2 一般性的遊戲規則

1. LATEX 的指令都是大小寫有別的，由 \ 開頭，後接由字母組成的字串或單一的非字母字元。其中由 [ ] 中括號括住的是選擇性參數，可以省略，由 { } 大括號括住的是不能省略的參數，當然，LATEX 的指令不一定會有參數，但絕大部份都會有參數，只不過把他給省略使用預設值罷了。

2. LATEX 文稿中，空一個英文空白和空多個英文空白的作用是一樣，LATEX 會認作一個英文空白。

3. 平常我們編輯純文字檔，按個 Enter 鍵，就代表換行，但實際排版出來，一行的寬度是按照排版版面的設定，也就是說，你在文稿中按 Enter，不代表排版後就是從這裡斷行，LATEX 會依一行應有的寬度經過整體計算後自動補成一整行後再來斷行，而且會在中間自動補足一個空白。這在英文很自然，稱爲字（word）間空白，但中文則不一樣，在編輯器中編輯中文，隨意按 Enter 的結果，會造成文章中的中文間出現空白。這會在本文中適當的時機，提出解決的方法。

4. 編輯器中，多按幾次 Enter 就多空出幾行，但在 LATEX 文稿裡，多個空白行，和一個空白行是一樣的作用，LATEX 會把他認作是一個空白行。而這個空白行，LATEX 同時也會認作是新段落的開始，所以 LATEX 是以空白行來分隔各個段落。

5. LATEX 預設每個章節的第一個段落的第一行是不內縮（noindent），從第二個段落開始才會內縮（indent）。當然，這是可以更改的，往後會再提及。

6. LATEX 的指令，是從反斜線後第一個字母開始，到第一個非字母符號爲止（包括空白、標點符號及數字）。因此：

```
This is my first \LaTeX typesetting example.
```

這樣的話，實際結果，因爲 \LaTeX 後的空白是屬於指令的一部份，空白將不會被解釋，這樣會印成：

This is my first LATEXtypesetting example.

這種結果，LATEX 和 typesetting 連在一起了。要避免的話，就要指定指令的作用範圍，例如以下的大括號。或就眞的加個空白，例如 \ ，LATEX 碰到 \ 就會形成完整的指令，其後的空白就會被眞正解釋爲空白了：

```
This is my first {\LaTeX} typesetting example.
This is my first \LaTeX{} typesetting example.
This is my first \LaTeX{} typesetting example.
```

所以，正常印出來應該是：

This is my first LaTeX typesetting example.

7. 註解符號（%），可以放在一行的任何地方，% 後的文字會被 LaTeX 忽略。所以，如果是放在一行的最尾端，那麼 LaTeX 會自動插入的字間空白也將會被忽略。例如：

```
This is my fisrt \LaTeX{} document. Give \LaTeX{} a%
try.
```

這樣一來，排版出來會變成：

This is my fisrt LaTeX document. Give LaTeX atry.

a 和 try 連在一起了！正常應該是：

This is my fisrt LaTeX document. Give LaTeX a try.

基於這個特性，我們可以應用在中文，也就是說在編輯器中，中文文章按 Enter 鍵換行時，尾端加個%，這樣一來 LaTeX 就不會插入英文字間空白，中文字就可以連成中間沒有空白的一整行了，否則 LaTeX 在整篇文稿斷句時，會自動在原換行處填入一個英文空白，因爲，原始的 TEX/LATEX 是認不得中文的。

8. 中英文混合的時候，通常，英文字前後都會留個空白，以便和中文區隔開來，只是這個空白要多大，這就沒有固定的慣例，通常留個英文空白也是可以，要講究的話，等談到中文排版相關議題時再來討論，目前就養成習慣，英文單字前後留個英文空白。

3.3.3 針對標點符號的遊戲規則

1. 中英文的引號不一樣，這裡請特別注意，許多人常常搞錯。中、英文引號不管單雙都要分左右。英文的話，左邊的引號是 grave accent，是鍵盤左上方 Esc 或 F1 下方有波形號的那一個鍵；右邊的是 apostrophe，也就是鍵盤左邊 Enter 鍵隔壁的那個鍵。雙引號的情形是鍵入兩次的左單引號及兩次的右單引號，而不是用 " 這個一次完成兩個點的 ditto marks。所以，實際上在鍵入文稿時是：

```
Please press an `Esc' key.
Please press an 'Esc' key. 這是錯誤示範！
``This sentence.''
"This sentence." 這是錯誤示範！
```

排版出來的情形是：

Please press an ‘Esc’ key.
Please press an ’Esc’ key. 這是錯誤示範！
“This sentence.”
”This sentence.” 這是錯誤示範！

中文的話，我們是使用中文全形的「、」及『、』，在中國大陸則已改用和英文相同形狀的全形符號，但這在中文直排時會出問題，因此，中文的單、雙引號還是得維持我們目前使用的。

2. LATEX 會在英文文章的一個句子結束和另一個句子開始的中間，自動調整成較大一點的空白，這可以增加文章的易讀性。所謂一個句子結束，例如：句點（.）、問號（?）、驚嘆號（!）及冒號（:），這當然是指英文的半形標點符號，不是中文的全形標點符號。你可以注意一下上面所舉的例子，在 document. 和 Give 之間的空白會稍微大於其他英文單字間的空白。

   現在的問題是，如果這些標點符號後面不是另一個句子的開始的時候，LATEX 無法去判斷這種情形，這時得由我們自己自行判斷、處理了。例如英文縮寫字：

   ```
   I am Mr. Edward G.J. Lee, G.J. is a abbreviation of my name.
   I am Mr.~Edward G.J. Lee, G.J. is a abbreviation of my name.
   I am Mr.\ Edward G.J. Lee, G.J. is a abbreviation of my name.
   ```

   其中 `Mr.\ Edward` 的寫法，和 `Mr.~Edward` 幾乎是一樣的，都是強迫插入一個比較小的正常單字間空白，差別在於後者也另外表示不可以從這裡換行，通常用在人名的時候，讓他們不致中斷，一般在人名的排版，包括他的頭銜、職稱，是不中斷成兩行而分開的。而且整個文句較長的話，以後者較恰當，才不會因爲斷行被分成兩半，這個 符號也因此在 TEX 的專有名詞，就稱爲 tie，把他們綁住的意思。排版出來的時候會變成：

   I am Mr. Edward G.J. Lee, G.J. is an abbreviation of my name.
   I am Mr. Edward G.J. Lee, G.J. is an abbreviation of my name.

   請放大去仔細比較一下結果就知道了。第二行的才是正確的，Mr. 和 Edward 之間的空白是正常單字間空白，比第一行的句子結束空白要小一點點。其他有使用到縮寫字的場合，例如：‘Dr.’、‘etc.’、‘e.g.’、‘i.e.’、‘vs.’、‘Fig.’、‘cf.’、‘Mrs.’，這些都不是代表句子結束，所以，要插入一個正常空白。

那 G.J. 後面爲什麼沒有插入正常空白？那是因爲，J 是大寫的，這時 LaTeX 不會去誤認爲是句子結束，通常句子結束時的句點前的那個字母是小寫的。Well，有沒有覺得有點道理？:-)

等等，事情還沒有結束！Knuth 教授出了一道考題，如果句子的結束是 'Please see Appendix A.' 後面又還接有另一個句子。這時怎麼辦？由於，不會認爲是句子結束，因此會插入正常空白，但這正是句子結束呀！請暫時先記得，使用 `...Appendix A\null.`，或 `...Appendix A\@.`。這個說來有點話長，有機會再來探討，請記得 '`\null`' 和句點間是沒有空白的。例如：

```
Please see Appendix A. We will be there soon.
Please see Appendix A\null. We will be there soon.
```

排版出來的結果將會是（差異不明顯，請小心比較）：

Please see Appendix A. We will be there soon.
Please see Appendix A. We will be there soon.

如果，你現在閱讀的是 HTML 格式文件，有些例子如果無法明顯顯示出來，請改閱覽 PDF 版本。而且，如果你製作 PDF 格式時，字型沒有內嵌（本文的英數字是嵌入 Computer Modern Type1 字型），差異可能將會更不明顯。可試著使用 gv/gsview 去閱覽，然後調整成 Landscape，把句子尾部拉到邊緣的地方去就看得出來了。這在句子多的時候，這個空白也並非固定大小的，LaTeX 會視文章結構的需要做細微的調整。

3. 刪節號中文英也是不同，英文是三點，如果碰到句點的話，則是四點。中文的話是六點，碰到中文句點很容易就分得清楚。但是英文這個三點，不是就打個三個句點了事，這樣的點太密集，可以使用 `\ldots` 或 `\dots` 指令，例如：

```
I'm not a good man ..., but a good husband .... 錯誤示範！
I'm not a good \ldots\ man \ldots, but a good husband \ldots.
I'm not a good \dots\ man \dots, but a good husband \dots.
```

排版出的來結果是：

I'm not a good ... man ..., but a good husband .... 錯誤示範！
I'm not a good … man …, but a good husband ….
I'm not a good … man …, but a good husband ….

中文的刪節號是由兩個全形的三點所組成六點的，即：……，就是我們 Big-5 碼的 `0xa14b`（`U+2026`），但由於 Unicode 尚有一個 MIDLINE HORIZONTAL ELLIPSIS(`U+22EF`)，因此，有些軟體在解讀上有可能會不一樣，因爲我們的字型，大部分在製作標點符號時是置在中央的地方，不像中國大陸是置放在基線的地方，

而 Unicode 官方採用的樣本字型，剛好是中國大陸的廠商所提供，這樣一來有些軟體工作者就認爲我們的刪節號應該是 U+22EF 了，很不幸的，我們的 Big-5 碼並沒有相對應的字碼。

4. 破折號。在英文，相當於破折號的可能有三種：

   - hyphen
     這是最短的 dash，通常就是鍵入 - 就行了，例如 father-in-low，這樣會表現成 fater-in-low。
   - en-dash
     這是最常用的破折號，是鍵入兩個 hyphen。例如 1991--2003 年，這會表現成 1991–2003 年。
   - em-dash
     這是最長的 dash，由三個連續的 hyphen 組成，應該是最相近於我們中文所說的破折號。例如 I am---a good man. 會表現成 I am—a good man.。至於這個三個連續的 hyphen 前後是否要留空白，都有人使用，並沒有硬性的慣例，但爲了和中文的破折號配合（中文破折號前後，通常不留空白），個人通常是不留空白的。
   - 眞正的減號
     這應不能算是破折號，而是實際的減號或負號，這要進入數學模式，例如：負五，要寫成 -5，然後表現出來是 -5。這也常常會有人搞錯，不能直接鍵入一般的負號那個鍵來充數，這是因爲 TEX/LATEX 的數學式子的用字和間隔處理，和一般內文不同的關係。
   - 中文的破折號
     中文的破折號是佔兩個中文字位置的的一橫線，長度和刪節號相同。在中線位置的，定義上是有兩種，en-dash 是 Big-5 0xa156，em-dash 是 Big-5 0xa158。但由於中文字間距的問題，有可能打出來的破折號中間會有一點空白[2]，例如——中文的 em-dash，這是——中文的 en-dash。在論文中，破折號通常可以使用小括號或冒號代替。
   - 中文的私名號及書名號
     中文的私名號，可以標明人名、地名，如孫逸仙；書名號（私名號的底線換成

[2]這是可以調整的，也就是去除兩個橫線之間的字間距，這樣就不會產生小空白了，中文刪節號也有同樣的情形，我們會在微調的部份再來討論。

波紋形狀），可以用在書名，這些符號常造成排版上的困擾，常使用《》來取代書名號，私名號則無其他取代方法。在一般的自然及應用科學論文上通常不使用這種舊式的私名號及書名號。

5. 避頭點

這可是排版的重要功能。英文的通常沒有問題，LATEX 會自動避開處理，中文就不一定了，LATEX 可不認識中文，但通常中文相關程式及套件，多多少少都會處理，只不過，有時候偶爾可能會誤判。那麼，到底什麼是避頭點？底下列個表，大家就明白了，我列中文的，英文的就不列出來了，因爲 LATEX 會自動處理，不必我們擔心。

標點符號	置放處
，。；、：」）》！？	不能置於行首
「（《	不能置於行尾
破折號及刪節號	置於首尾皆可

簡單的說，除了破折號及刪節號，沒有開口的，不能置於最開頭，開口向右的，不能置於最右，開口向左的，不能置於最左。通常都會處理好，但校稿的時候要注意一下誤判的地方。

3.4 LATEX 的文稿結構

3.4.1 環境（environment）

上一節所談的都是指令，雖然也可以由大括號來定作用範圍，但如果是一整段，甚至是一整篇文章都要作用時，那指令可能就不很適合了，因此，LATEX 也有一種巨集結構，稱爲環境（environment），主要是讓作用範圍能擴大至較大的範圍。

所有的環境，都是起於 `\begin{環境名稱}`，止於 `\end{環境名稱}`，這兩個指令之間的文稿都會被作用，而且，環境之內還可以套用其他不同的環境。

LATEX 文稿的內文，其實就是包在一個 `\begin{document}` 和 `\end{document}` 這個 document 環境當中。

3.4.2 最簡單的 LATEX 的文稿結構

以下就是所有 LATEX 必需具備的文稿大結構：

```
\documentclass{article}

  這裡是 preamble 區

\begin{document}

  這裡是本文區

\end{document}
```

\documentclass{article}，這是在告訴 LATEX 使用哪一種類別，我們目前使用的是 article 類別，關於類別會在第 6 章討論。preamble 區，則是下一些會影響整個文稿的指令，及引用巨集套件的地方，當然，完全不引用巨集，也不使用影響全文的指令的話，preamble 區就是空白，不寫任何東西。本文區，就是我們實際上寫文章的地方。

現在也可以把前面所舉的例子，放入本文區裡頭，preamble 區空白沒關係，然後存檔，試著編譯看看：

```
latex example.tex
dvips example.dvi      => 產生 ps 格式 example.ps
dvipdfm[x] example.dvi => 產生 pdf 格式
pdflatex example.tex   => 直接由 .tex 產生 .pdf
```

眞正的實例解說，會在下一章來進行，所以，這裡暫時不會介紹有什麼環境可以使用，先玩看看沒有關係。由於還沒談到中文的問題，因此如果你想試試看，那暫時先使用英文，道理都是相通的。

3.4.3 preamble 區可以放些什麼？

這裡可以引用巨集，而且會影響整篇文稿的指令，例如一些事先定義好的指令，想在整篇文稿中使用，就可以置放在 preamble 區。

3.4.3.1 巨集的引用

本文主要是標準 LATEX，但前面已提到，會有些巨集套件不得不要引用，底下就來說明如格引用套件。這些套件都是一般 TEX 系統都會附上的。

指令及環境要如何開頭都介紹過了，現在來看看引用巨集要怎麼開頭。

```
\documentclass{article}
```

```
\usepackage{color}
\begin{document}
\textcolor{blue}{This is blue color.}
\end{document}
```

編譯一下，看看結果是什麼？這裡使用的就是 color package，裡頭是由 TEX/LATEX macro 所寫成一個巨集套件。一般簡單的我們就稱爲巨集（macro），複雜一點的就稱爲巨集套件（package），其實，裡頭都是一樣的，只不過大小及有沒有整理成一個系統的差別。

LATEX 裡頭有什麼現成的套件可以使用，每個散佈的 TEX 系統所收集的可能都會有所不同。大概沒有人可以精通所有現存的 LATEX 巨集套件，因爲實在是太多了，不過，本文大概都會提到常用的巨集套件。詳細的巨集套件的種種，會在第 7 章來說明。

3.4.3.2 影響整篇文稿的指令

會影響整篇文稿的指令，通常也是放在 preamble 區，例如：

```
\linespread{1.36}
\parindent=0pt
```

\linespread 是在控制上下行的行距，這裡就是將行距變成原來的 1.36 倍。至於什麼是行距呢？就是這一行的基線（baseline）到下一行的基線的距離，通常英文文章不必去調整他的行距，但中文得適當加大行距以利閱讀。

\parindent 是調整段落內縮的程度，這裡調整成 0，也就是說各段落都不內縮的意思，也可以調整成其他的值，LATEX 就會依這個值去內縮。當然，不去設定的話，LATEX 就會依他的預設值去內縮。

3.4.4 章節結構

本文區當然是我們寫文章的主要地方，及一些微調。在 LATEX 的文稿裡頭，章節標題的形成都是由同樣的指令來控制的，這樣有一個好處，臨時插入章節標題及其內文時，我們不必去理會標題編號及目錄的問題，也不必去理會要用什麼字型、及字型大小要多大，LATEX 會自動計算處理，字型大小也會和內文使用的字型大小互相配合調整，使用者就專心在內文構思、寫作即可。以下由列表來瞭解整個章節結構：

深度標號	指令	作用及注意事項
−1	`\part{}`	這是最大的結構，我們中文通常稱爲「部」。
0	`\chapter{}`	章。在 article 類別裡頭沒有章。
1	`\section{}`	節。
2	`\subsection{}`	小節。
3	`\subsubsection{}`	次小節。
4	`\paragraph{}`	段落。
5	`\subparagraph{}`	小段落。

章節標題的內容就是直接寫入指令的大括號裡頭就可以了，LATEX 在排版時會自動使用粗體、加入章節編號及納入目錄裡頭。

至於第一欄的深度標號（secnumdepth），`book/report` 類別的深度標號是 2，`article` 的是 3。這是什麼意思呢？就是說 `book/report` 類別的文稿，在 `\subsection{}` 以後（`subsection` 本身仍會編號），章節就不再編號了；同樣的，在 `article` 類別的文稿，在 `\subsubsection{}` 以後就不編號了。但仍然會獨立出一單獨行來表示這個是標題。不編號了的章節內容，當然也就不納入目錄裡頭了。這當然是可以更改的，只要更改 LATEX 的 `secnumdepth` 這個變數的值就可以了，這個往後會提及如何更改 LATEX 的預設值。像這篇文章，在 preamble 區就有一個設定：

```
% let the depth of report to subsubsection
\setcounter{secnumdepth}{3}
```

所以，這篇文章雖然使用的是 `report` 類別，但是章節的深度標號是標在 3，也就是說會編號到 `subsubsection` 爲止，但這仍然是沒有編入目錄中的。

下一章就讓我們開始實際動手吧！但 ……，怎麼到現在都沒有完整介紹指令呢？那我怎麼會知道有什麼指令可以使用？這是因爲 LATEX 的指令很多，直接介紹的話，一方面記不住，二方面也不容易瞭解他的實際作用，所以，我們將會在下章舉例時穿插在裡頭說明，等這份文件接近尾聲時，再來整理個指令速查表，這樣以後查指令就很方便了，不必去死記，只要知道有個這樣功能的指令就夠了。

第 4 章

實際上排版玩看看

好了，現在正式來玩看看吧！本章主要是簡單的實例說明，先進入狀況再談其他。實際應用比高談闊論有用多了。

先來個「最高指導原則」：**學會控制空間，你就學會排版了**！剛學排版的朋友，往往會把所學到的東西去想法子佈滿你所有的空間（就是一個頁面），但實際上，你要調整的，其實是各個部份的空間配置，抽象一點說，也就是一整個頁面當中，沒有文字、圖表的部份才是排版眞正的重點。你先聽聽就好，過一段時間的熟悉後，再來回頭思考這個「玩弄」空間的原則。:-)

由於本文有 HTML 版本，爲了轉換上不造成失眞，實例的部份，文稿上只寫程式碼，結果的部份都是編譯好的 PDF 格式檔案，置放於網站上，可線上閱覽或下載，簡單的例子則不另製作獨立的 PDF 檔，請參閱本文的 PDF 格式檔案內容。

4.1 簡單的實例

這裡就把前一章所談到的一些內容整理成一個文稿，先來試試看，這裡先使用 `report` 類別文稿，因爲 `article` 類別文稿是沒有 `chapter` 的：

```
% example1.tex
\documentclass{report}
\begin{document}
This is my first {\LaTeX} typesetting example.\\
This is my first \LaTeX{} typesetting example.\\
This is my first \LaTeX{} typesetting example.\\
I am Mr. Edward G.J. Lee, G.J. is a abbreviation of my name.\\
I am Mr.\ Edward G.J. Lee, G.J. is a abbreviation of my name.\\
Please see Appendix A. We will be there soon.\\
Please see Appendix A\null. We will be there soon.
\end{document}
```

使用編輯器編輯，然後存檔成 exmaple1.tex，這樣就可以編譯了：

```
latex example1.tex       => 產生 example1.dvi
dvips -Ppdf example1.dvi => 產生 example1.ps
ps2pdf example1.ps       => 產生 example1.pdf 或
dvipdfm[x] example1.dvi  => 由 example1.dvi 直接產生 example1.pdf 或
pdflatex example1.tex    => 由 example1.tex 直接產生 example1.pdf
```

編譯好的 PDF 檔可在此下載或閱覽：

http://edt1023.sayya.org/tex/latex123/example1.tex
http://edt1023.sayya.org/tex/latex123/example1.pdf

4.1.1 關於換行

每行最後加了個 \\，這表示強迫換行的意思，否則 LaTeX 會依版面預設的寬度來換行，就不會是一個句子一行了，大家可以把這個 \\ 拿掉，再來編譯試看看結果，就會知道怎麼一回事了。也可以使用 \newline 這個指令，當然，我們都會聰明的選用較短的指令。而且 \\ 可以控制換行時的間隔，這在 \newline 則不行。例如：

```
Please see Appendix A. We will be there soon.\\[1cm]
Please see Appendix A\null. We will be there soon.
```

這樣的話，兩行之間的行距就是原來的行距再加上 1cm。甚至，也可以是負數的參數，這樣行距就會變成原來的行距減去 1cm，當然，如果設過頭了的話，兩行可能會重疊在一起。既然，這裡使用的是方括號，表示這些參數是可以省略的。

另外，\linebreak[n] 也可以強迫換行，n 代表由 1–4 的建議值，數值愈大表示愈是強烈建議，不設定的話，就是換或不換兩種選擇，沒有中間地帶。和前面所說的不同處是，這種換行會把原來那一行句子的長度平均布滿版面上行寬的長度。例如：

```
Please see Appendix A. We will be there soon.\linebreak
Please see Appendix A\null. We will be there soon.
```

排版後會表現成：

Please see Appendix A. We will be there soon.
Please see Appendix A. We will be there soon.

4.1.2 關於縮排

第一行縮排了！這是因爲我們完全沒有分章節，所以，LaTeX 就把這些內容當做是引言的部份，依 LaTeX 的安排，引言開頭是會縮排的。要解決這個問題，可有兩種方法：

1. 在第一行之前加入 `\noindent` 來指示 LaTeX 不要去縮排。但是這只作用在下指令的地方，其他該縮排的地方還是會縮排。
2. 在 preamble 區加入 `\parindent=0pt`，這表示讓全文的縮排爲 `0pt`，當然，這就表示全文都不要縮排了。

4.2 加入章節標題

在 LaTeX 裡頭，要加入章節標題實在是太容易了，也不必去管字體的大小及置放的位置，盡管加上去就對了！LaTeX 會替我們安排一切。我們這裡仍然以 `report` 類別來說明，因爲 `article` 類別裡頭，沒有章，只能適用於較簡單的短文。

```
% example2.tex
\documentclass{report}
\begin{document}
This is the first experience of \LaTeX.
\chapter{Aesop Fables}
\section{The Ant and the Dove}
An ant went to the bank of a river to quench its thirst, and
being carried away by the rush of the stream, was on the
point of drowning.
A Dove sitting on a tree overhanging the water plucked a
leaf and let it fall into the stream close to her. The Ant
climbed onto it and floated in safety to the bank.
\section{The Dog in the Manger}
A dog lay in a manger, and by his growling and snapping
prevented the oxen from eating the hay which had been
placed for them.
``What a selfish Dog!'' said one of them to his companions;
``he cannot eat the hay himself, and yet refuses to allow
those to eat who can.''
\chapter{The Eagle and the Arrow}
An eagle sat on a lofty rock, watching the movements of a
Hare whom he sought to make his prey.
An archer, who saw the Eagle from a place of concealment,
took an accurate aim and wounded him mortally.
\end{document}
```

編譯出來的結果：

http://edt1023.sayya.org/tex/latex123/example2.tex
http://edt1023.sayya.org/tex/latex123/example2.pdf

請注意他什麼時候會縮排，什麼時候會換頁。report 類別，新的一章會換頁，如果想節省一點空間，可以換用 article 類別，\chapter{} 改用 \section{}，原來 \section{} 就改用 \subsection{}，這樣就不會換頁，內容就會連續下去了。大家可以試著把 report 改成 article 及 book 再重新編譯一次，試試看結果有何不同。

4.3 加入 title page 資訊

這是指內頁的第一頁，我也不知道這個中文專有名詞是什麼，在 LaTeX 裡頭，我們就稱爲 title page。在 LaTeX 的標準格式裡，他包括了標題（title）、作者名字（author）、日期（date）及感謝詞（thanks）。要注意的是，在 report/book 類別，title page 是自成一單獨頁的，但在 article 類別裡，他是和本文連起來的。我們就以上面的伊索寓言的文章爲例，要修改的地方是 preamble 區及本文區的 \maketitle：

```
% example3.tex
\documentclass{report}
\title{Aesop Fables}
\author{Aesop\thanks{Thanks to the reader.}
      \and Nobody\thanks{Thanks to nobody.}}
\date{\today}
\begin{document}
\maketitle
This is the first experience of \LaTeX.
\chapter{Aesop Fables}
\section{The Ant and the Dove}
 ...
```

排版出來的結果如下：

http://edt1023.sayya.org/tex/latex123/example3.tex
http://edt1023.sayya.org/tex/latex123/example3.pdf

我們可以發現，這一頁是不編頁碼的，從下一頁開始才是第一頁。作者可以有多個，使用 \and 指令來連接。日期不一定要有，如果沒有 \date{\today} 這個指令，那還是有日期，但只能固定在今天。如果內容過長，他會自動折行，但也可以手動加 \\ 來強迫換行，不管如何換行，整個句子是居中排列的。\maketitle 是下在本文區的開頭，如果不下這個指令，那編譯時不會有什麼錯誤，只是就沒有 title page 了。

4.4 加入目錄（Table of Contents）

加入目錄（Table of Contents）對 LaTeX 而言，更是輕而易舉的事情，只要在本文開頭加個 \tableofcontents 指令就成了！依上面的例子，修改成：

```
% example4.tex
\documentclass{report}
\title{Aesop Fables}
\author{Aesop\thanks{Thanks to the reader.}
      \and Nobody\thanks{Thanks to nobody}}
\date{\today}
\begin{document}
\maketitle
\tableofcontents
This is the first experience of \LaTeX.
\chapter{Aesop Fables}
\section{The Ant and the Dove}
 ...
```

排版出來的結果如下：

http://edt1023.sayya.org/tex/latex123/example4.tex
http://edt1023.sayya.org/tex/latex123/example4.pdf

這裡千萬要注意的是，\tableofcontents 要加在 \maketitle 的後面，否則目錄會印在 title page 之前。而且要**編譯兩次**。第一次產生 example4.toc，然後第二次編譯再跟據這個 toc 檔，眞正編入目錄。

目錄是包括圖表目錄的（List of Figures, List of Tables），但我們目前還沒有談到圖表的排版，因此暫時略過，等談到時再來看要如何加入圖表目錄。

4.5 加入摘要（abstract）

這不一定會有，如果要加入的話，可使用 abstract 環境，在這個環境中的文章，左右會縮排。要注意的是，只有 article/report 類別才有 abstract，book 類別不能使用這個環境。

```
% example5.tex
\documentclass{report}
\title{Aesop Fables}
\author{Aesop\thanks{Thanks to the reader.}
```

```
      \and Nobody\thanks{Thanks to nobody}}
\date{\today}
\begin{document}
\maketitle
\begin{abstract}
The tale, the Parable, and the Fable are all common and popular
modes of conveying instruction. Each is distinguished by its own
special characteristics.
\end{abstract}
\tableofcontents
\chapter{Aesop Fables}
\section{The Ant and the Dove}
 ...
```

排版出來的結果如下：

http://edt1023.sayya.org/tex/latex123/example5.tex
http://edt1023.sayya.org/tex/latex123/example5.pdf

report 類別的摘要自成一頁，不編頁碼，且不會編入目錄中，這和一般的論文格式可能會不一樣，使用時請注意。artcile 的類別則仍然是和本文相連的，會出現在文章標題之後。

abstract 和 summary 在較正式的論文是有區分的，通常 abstract 在文前；summary 則在文後。但目前一般性的文章則沒有這樣區別，通通當成「摘要」。通常，摘要裡頭是不用註解、無交互參照也不使用公式圖表的。

4.6 加入註解

在 LaTeX 裡頭，註解可有兩種方式，一種是腳註（footnote），一種是邊註（marginal note）。通常 LaTeX 的腳註預設是由阿拉伯數字在編號，置於頁底部。在沒有部（part）的情形下，report/book 類別，編號每章會從頭起算，article 類別則會連續，而且，會使用 footnotesize 的字體印出。邊註則不編號，字體是正常大小。

4.6.1 腳註（Footnote）

在所要加註的那個字後，使用 \footnote{} 指令即可，解說的文字就寫入大括號之內，一般 LaTeX 的指令在此都仍然有有作用，會印在此頁的底部，以小一點的字來印出，並加上

編號。以下我們就試試看在 Dove 這個字來做腳註。請注意，Dove 這個字和 \footnote{} 之間是沒有空白的。

```
% example6.tex
\documentclass{report}
\title{Aesop Fables}
\author{Aesop\thanks{Thanks to the reader.}
      \and Nobody\thanks{Thanks to nobody}}
\date{\today}
\begin{document}
\maketitle
\tableofcontents
This is the first experience of \LaTeX.
\chapter{Aesop Fables}
\section{The Ant and the Dove}
An ant went to the bank of a river to quench its thirst, and
being carried away by the rush of the stream, was on the
point of drowning.
A Dove\footnote{Pigeon, an emblem of peace.}
sitting on a tree overhanging the water plucked a
leaf and let it fall into the stream close to her. The Ant
climbed onto it and floated in safety to the bank.
 ...
```

排版出來的結果如下：

http://edt1023.sayya.org/tex/latex123/example6.tex
http://edt1023.sayya.org/tex/latex123/example6.pdf

4.6.2 邊註（Marginal note）

邊註只是把 \footnote{} 換成 \marginpar{} 而已，內容仍然寫入大括號內。但和腳註不一樣的是，他沒有編號（因爲就在旁邊，無此必要），他的字體也不會小一號，和內文的字體大小是一樣的，這在後面討論到字型的時候會談到如何改變字體的大小。

```
% example7.tex
\documentclass{report}
\title{Aesop Fables}
\author{Aesop\thanks{Thanks to the reader.}
      \and Nobody\thanks{Thanks to nobody}}
\date{\today}
\begin{document}
\maketitle
\tableofcontents
This is the first experience of \LaTeX.
```

```
\chapter{Aesop Fables}
\section{The Ant and the Dove}
An ant went to the bank of a river to quench its thirst, and
being carried away by the rush of the stream, was on the
point of drowning.
A Dove\marginpar{Pigeon, an emblem of peace.}
sitting on a tree overhanging the water plucked a
leaf and let it fall into the stream close to her. The Ant
climbed onto it and floated in safety to the bank.
 ...
```

排版出來的結果如下：

http://edt1023.sayya.org/tex/latex123/example7.tex
http://edt1023.sayya.org/tex/latex123/example7.pdf

4.7 字型的相關調整

TEX/LATEX 的字型系統算是相當複雜的，這裡不多談其中原理，站在使用者的角度，我們只要知道怎麼使用就行了。在這裡，我們說字型（font），指的是字型本身的一個總稱，或稱爲字體，在字的形狀的時候，我們就稱爲字形（font shape）。

LATEX 使用的字型選字機制，以目前新版本的 LATEX 而言，是使用 1993 年發行的 NFSS(New Font Selection Scheme) 第二版爲標準。當然，仍然是建立在 TEX 字型機制的基礎上的，這已超出這篇文章的範圍。

4.7.1 LATEX 對字型的屬性描述

在 LATEX 裡，對於字型的描述，使用了五種屬性來說明，這五種屬性，也是 LATEX 巨集中常要使用到的參數，甚至是錯誤訊息標示字型來源的時候，會把字型的這些屬性給顯示出來。

1. 字型編碼（font encoding）
   這裡所謂的字型編碼，指的是各個個別的字在一個字型裡頭的排列順序及安排方式。原始的 TEX 字型編碼我們就稱爲 OT1(Old TEX text encoding)，這是預設的，如果都不指定字型編碼，那所使用的就是 OT1 編碼。在目前新一代的字型編碼裡頭，字

的安排方式及內容和 OT1 不一樣，例如 T1[1]，這在往後提到改變字型編碼時會再談到，我們目前就不去調整字型編碼，使用預設的 OT1，其他的編碼這裡就不多談了。

2. 字族（font family）
   指同一設計類型的字型集合的名稱，例如羅馬字族（roman）、打字機字族（typewriter）等等，通常前面會冠上製作商或製作人的名稱，例如 Knuth 教授設計的，稱爲 'Computer Modern Roman'，Adobe 公司製作的羅馬字族稱爲 'Adobe Times'。我們預設使用的，當然就是 Knuth 教授所設計的 Computer Modern fonts。以下爲一些例子：

簡稱	代表意義
`cmr`	Computer Modern Roman
`cmss`	Computer Modern Sans Serif
`cmtt`	Computer Modern Typewriter

3. 字型系列（font series）
   這是指字型的 weight（胖瘦）及 width（長扁）來區分的。例如粗、細字體，一般我們正常用的是 medium，粗體則是 bold。以下是一些例子：

簡稱	代表意義
`m`	medium
`b`	bold
`bx`	Bold extended
`sb`	Semi-bold
`c`	Condensed

4. 字形（font shape）
   這個望文生義，就是字的形狀。例如意大利斜體（italic）、斜體（slant）、small caps 等等。以下是幾個例子：

簡稱	代表意義
`n`	正常字（normal），指 upright 或 roman
`it`	Italic
`sl`	Slanted
`sc`	Small Caps

[1]正式名稱是 Cork's TEX extended text encoding 又稱爲 Text Companion encoding。這裡的 T1 和 Type 1 字型規格無關，他是字型編碼方式，他把字型裡頭有關一些重音符號字母單獨視爲一個單獨的字，而非如 OT1 是由一般字母和重音符號組合而成。

5. 字型大小（font size）
   預設的字型大小是 10pt（10 point），十點字。不加單位的話，預設的就是 pt。請注意，非標準 LaTeX 類別的預設字型大小可能會不一樣。

我們對字型要調整改變的，就是這些字型屬性的設定值。LaTeX 已設定好方便的指令給我們使用。

4.7.2 調整字族、字型系列、字形的指令

	字型	標準指令	宣告式指令（環境）	舊用法
字	textup	`\textup{textup}`	`{\upshape textup}`	
形	*italic*	`\textit{italic}`	`{\itshape italic}`	`{\it italic}`
	slant	`\textsl{slant}`	`{\slshape slant}`	`{\sl slant}`
	SMALL CAPS	`\textsc{small caps}`	`{\slshape small caps}`	`{\sc small caps}`
系	medium	`\textmd{medium}`	`{\mdseries medium}`	
列	**boldface**	`\textbf{boldface}`	`{\bfseries boldface}`	`{\bf boldface}`
字	roman	`\textrm{roman}`	`{\rmfamily roman}`	`{\rm roman}`
族	sans serif	`\textsf{sans serif}`	`{\sffamily sans serif}`	`{\sffamily sans serif}`
	typewriter	`\texttt{typewriter}`	`{\ttfamilyfamily typewriter}`	`{\ttfamily typewriter}`

先別嚇了一跳，這是有跡可循的。其中 upright, medium, roman 都是一樣的，這是一般的正常字，就不必麻煩去設定他了，除非是要在特定字型範圍裡頭，重新改變成正常字體。從前面所說的簡稱的字串，再和 text, family, series, shape 去配對來使用，這樣只要記得簡稱就行了，例如：italic 的就是 `\textit{}`。不然也可以使用「偷吃步」的舊用法，其實這也不是什麼偷吃步，他是原始 Plain TeX 所定義的，在舊版的 LaTeX 2.09 也相容他而沿用，並加以擴充。但如果使用舊用法，那有時組合式的表示時可能會無效，例如粗斜體這種粗體和斜體設定混合時，就無法產生粗斜體了，這時還是得乖乖使用正統標準 LaTeX 的表示法。

要注意的是，大括號的位置，宣告式的指令，整個作用範圍是連指令一起包住的，他可以當成環境來使用，例如 `\begin{itsahpe}`, `\end{itshape}`，這樣在這個環境內的文字就通通會使用 italic 斜體，也可以不加參數使用，例如 `\itshape`，這樣以下的文字通通會使用 italic 斜體，直至另一個改變字型的指令出現爲止。標準指令的作用範圍則是當做指令的一個參數，這些參數是出現在指令後的大括號內的。現在就來實際編譯個例子試看看：

```
% example8.tex
\documentclass{report}
```

```
\title{\bfseries Aesop Fables}
\author{Aesop\thanks{Thanks to the reader.}
      \and Nobody\thanks{Thanks to nobody}}
\date{\today}
\begin{document}
\maketitle
\tableofcontents
\chapter{Aesop Fables}
\section{The \textsl{Ant} and the \textsl{Dove}}
\itshape
An antwent to the bank of a river to quench its thirst, and
being carried away by the rush of the stream, was on the
point of drowning.
\upshape
A \textsl{Dove} sitting on a tree overhanging the water plucked a
leaf and let it fall into the stream close to her. The \textbf{\textsl{Ant}}
climbed onto it and floated in safety to the bank.
\section{The {\it Dog}\/ in the Manger}
A \textbf{\textit{dog}} lay in a manger, and by his growling and snapping
prevented the oxen from eating the hay which had been
placed for them.
``What a selfish Dog!'' said one of them to his companions;
``he cannot eat the hay himself, and yet refuses to allow
those to eat who can.''
\chapter{The \textsc{Eagle} and the Arrow}
An \textsc{eagle} sat on a lofty rock, watching the movements of a
Hare whom he sought to make his prey.
An archer, who saw the \textsc{Eagle} from a place of concealment,
took an accurate aim and wounded him mortally.
\end{document}
```

我們把 title page 的標題改成粗體（請注意，宣告式或舊用法，大括號是把指令和文字整個括住的），把 Dove 改成 slant 斜體，把 dog 改成 italic 粗斜體[2]，把 ant 改成 slant 粗斜體，把 eagle 改成 small caps。由於章節標題原本就會轉換成粗體，所以章節標題的部份，粗體就不必重複設了。

但這裡發現例子裡第二章標題中的 Eagle 並沒有改變字體，而且以 `latex` 編譯時會產生以下的錯誤（這些訊息也會在 `example8.log` 中找到）：

```
  ...
LaTeX Font Warning: Font shape `OT1/cmr/bx/sc' undefined
(Font)              using `OT1/cmr/bx/n' instead on input line 4.
  ...
LaTeX Font Warning: Some font shapes were not available, defaults substituted.
```

[2]請注意，這兩種斜體是不一樣的，slant 是一般正常的字，只是把他傾斜個角度而已，但 italic 則是另一種獨特的字型設計。

...

現在我們看到了前面所談的屬性簡稱，這在 LaTeX 就會使用這種屬性來表示而發出訊息，這裡 `OT1/cmr/bx/sc` 就表示了 OT1 編碼，Computer Modern Roman 字族，Bold extended 系列，而且是 small caps 形狀的字型，錯誤訊息顯示，他並沒有定義，因此，這個字型將會使用預設的字型來代替，這裡就是以 n 正常形狀的 bx 系列字型來替代。所以，字型指令並不是都可以隨意組合的，有些是根本就沒有這種字型，有些則是沒有用巨集去定義好，這樣 LaTeX 就取不到字了，但別擔心，頂多就是使用預設的字型罷了！

另一個很奇怪的地方，就是第一章、第二節的標題，爲什麼是 `{\it Dog}\/ in the...}`？這個插入的 `\/` 是什麼東西？這是 TeX 系統調整斜體字（包括 iatlic 及 slanted）和正常字之間的空白的一個指令，稱爲 italic correction。這樣，在斜體字和正常字之間的空白才會正常。那爲什麼其他的斜體指令沒有加這個調整呢？這是因爲 LaTeX 巨集在設計時就有考慮到這個問題，所以 `\textit{}` 這類標準指令都會自動調整 italic correction，不必由我們手動調整。

另外，章節標題本就會自動轉換成粗體，標題上的 dog 爲什麼沒有變粗體？這在前面有提到過，這種舊用法有時是無法複合使用的，粗體又斜體的指令會用不上來。因此，建議盡量使用 LaTeX 的第一種標準指令來改變字型。使用 `{\it ...}` 或 `{\itshape ...}`[3] 這種指令的話，就得時時注意 italic correction 的問題，也得注意是否可以複合使用指令的問題，所以，還是不要偷懶的好。:-)

後下是排版出來的結果：

http://edt1023.sayya.org/tex/latex123/example8.tex
http://edt1023.sayya.org/tex/latex123/example8.pdf

4.7.3 相對字型大小的調整

接下來談最後一個字型大小屬性的調整，這在使用上比較單純，只要知道指令就可以馬上拿來使用。但是 TeX/LaTeX 系統中，談到字型，裡頭一堆地雷，例如前面談到正常的內文字型大小是 10pt，現在如果想製作海報，需要 64pt 的字的時候就會發現，設不出來了！正常 LaTeX 的定義，字型的大小範圍是在 5–24.88pt 之間，超出這個範圍的字需要其他的 package 的幫忙。[4]

[3] 宣告式指令可以複合使用，但仍然會需要手動做 italic correction。

[4] 這是 LaTeX 本身巨集定義的問題，因爲他主要是針對一般性文件及書籍，TeX 本身的能力，可以讓字型放大到 2047pt。

這裡我們先來看看內文 10pt 時各種字型大小指令、實際例子及其大小（這是相對大小，會隨內文預設字型大小而自動調整）：[5]

指令	實際例子	效果	實際的大小（點數）
\tiny	{\tiny tiny}	tiny	5pt
\scriptsize	{\scriptsize scriptsize}	scriptsize	7pt
\footnotesize	{\footnotesize footnotesize}	footnotesize	8pt
\small	{\small small}	small	9pt
\normalsize	{\normalsize normalsize}	normalsize	10pt
\large	{\large large}	large	12pt
\Large	{\Large Large}	Large	14.4pt
\LARGE	{\LARGE LARGE}	LARGE	17.28pt
\huge	{\huge huge}	huge	20.74pt
\Huge	{\Huge Huge}	Huge	24.88pt

這些字型大小指令也可以當成環境來使用，例如：

```
\begin{small}
  本文內容
\end{small}
```

這樣用也是可以的。

4.7.4 絕對字型大小的調整

通常字型的大小，使用上一節所說的相對字型大小來調整會比較方便，而且對於整個版面的配合也會比較恰當，例如行距也會跟著做適當的調整，如果自行用絕對字型大小的方法來調整字型大小的話，常常會造成行距不一致的情形，因此，如非必要，應盡量避免。

但有時候就是需要做這樣的調整，例如本文封面的字型大小，縱使是 LaTeX 預設的最大字型也覺得稍小了點，這就要另外引入 package 調整了。

這裡我們使用 type1cm package 來調整。當然，得使用 Type 1 字型，才可以達到無段放大、縮小的目的[6]，而這個 package 也是配合 Type 1 字型使用的。個別放大的 pk 點陣字，這裡就不討論了，目前絕大部份的 Computer Modern 字型都已有 Type 1 的 free 版本，而且各個 TeX distribution 都會附上，使用上會較方便。以下是 type1cm 的使用方法：

...

[5]請注意，本文 pdf 格式內文使用 12pt 字型大小，列表及其中的例子，是由另外 10pt 預設字型大小所製作的 eps 圖檔引入，以免失眞。

[6]LaTeX 系統中的字型放大，在 10pt 以上，是以 1.2 的倍數爲次方來放大的，因此，正文 10pt 的字型大小的話，不會有 13pt 這種大小的字型，LaTeX 會選用最相近大小的字型來替代。

```
\usepackage{type1cm}
  ...
\fontsize{字型大小}{行距大小}\seclectfont
  ...
```

還記得如何引用巨集套件嗎？請參考第 3 章、第 3.4 節，第 3.4.3 小節的說明。

其中的「字型大小」就是所要指定的大小，通常以 pt 爲單位，當然，要使用其他單位也是可以。「行距大小」也是要一併指定，不可省略。最後的 \seclectfont 是讓他發生作用的意思，LaTeX 有些關於字型的較低階指令，要下 \seclectfont 後才會作用，\fontsize{}{} 正是其中之一。

4.8 原文照列

什麼是原文照列？一般 LaTeX 遇到倒斜線會認爲是一個指令的開始，如果連整個指令都要印出的時候呢？這時就要用到原文照列的指令及環境了。

4.8.1 原文照列指令

如果只是一小段的文字要原文照列，那使用指令會比較方便，這個指令就是 \verb|文字內容|，其中的 | 這個符號可以使用其他非字母的符號代替，只要前後相同就行了，例如：\verb+文字內容+ 這樣也是可以的。

4.8.2 原文照列環境

如果是一整段的內容要原文照列的話，使用環境會比較方便，那便是 verbatim 環境。不管是哪一種原文照列的情形，預設是使用打字機字族的字型來顯示的。底下是一個簡單的例子，說明原文照列指令及環境的使用：

這裡會發現一些奇怪現象，例如 \verb* 那個星號是什麼意思呢？就是讓空白以 ␣ 的方式表示出來的意思，verbatim 環境也是可以這樣使用。例如 example9 中的：

```
\verb*|This is    4 space here.|
```

也可以寫成：

差別在於，環境的上下行會多空出個空白行出來。

另外，標題爲什麼不使用 \verb|\verb| 就好了呢？原因是原文照列的指令和環境都不能當做其他指令的參數，標題本身就是一個指令，所以 \verb 不能在裡頭。

使用 \textbackslash 這麼長的敘述，而不用 $\backslash$ 這個簡單的方式，原因是這個文稿有使用 LaTeX2HTML 來轉成 HTML 格式，使用 LaTeX 的替代表示法會轉成一般的符號，但使用後者的方式則會轉成圖檔，所以這裡就使用 LaTeX 的替代表示法了。

底下是排版出來的結果：

http://edt1023.sayya.org/tex/latex123/example9.tex
http://edt1023.sayya.org/tex/latex123/example9.pdf

4.9 加入中文

這裡只說明如何使用 CJK package 的情形，原因是一般 TeX distribution 會附上（有些發行套件並沒有附上，這時只好自行安裝了）。CJK package 是把中文的部份包在一個環境裡頭，在這個環境內就可以使用中文，離開這個環境就又回復到原本的英文環境，底下由例子來說明。

```
\documentclass{article}
\usepackage{CJK}  % 使用 CJK 巨集套件
\begin{document}
% 進入 CJK 環境，並使用 Big-5 碼及 hwmm 這個字型
\begin{CJK}{Bg5}{hwmm}
\section{CJK 巨集套件}
這是一個測試，關於 CJK package 的測試。
\section{桃花源記節錄}
初狹，纔通人；復行數十步，豁然開朗。土地平曠，屋舍儼然。有良田、美池、%
桑、竹之屬，阡陌交通，雞犬相聞。其中往來種作，男女衣著，悉如外人；黃髮、%
垂髫， 並怡然自樂。見漁人，乃大驚，問所從來；具答之，便要還家，設酒、殺雞、%
作食。村中聞有此人，咸來問訊。自云：「先世避秦時亂，率妻子邑人來此絕境，%
不復出焉；遂與外人間隔。」問今是何世；乃不知有漢，無論魏、晉。此人一一%
爲具言所聞，皆歎惋。餘人各復延至其家，皆出酒食。停數日，辭去。此中人語%
云：「不足爲外人道也。」
\end{CJK}
\end{document}
```

就這麼簡單。只是編譯要改由 bg5latex 而不是原來的 latex 指令，這是爲了避開我們 Big-5 碼的一些特殊碼的關係，還記得爲何每行最後要加個百分號 % 嗎？這樣才不會插入英文的字間空白。編譯好的例子如下：

http://edt1023.sayya.org/tex/latex123/example10.tex
http://edt1023.sayya.org/tex/latex123/example10.pdf

詳細的 CJK package 的使用中文說明，請參考 CJK package 所附的文件及〈我的 CJK〉一文：

http://edt1023.sayya.org/tex/mycjk/mycjk.html
http://edt1023.sayya.org/tex/mycjk/mycjk.pdf

第 5 章

空間與位置

前一章曾提到過，學會控制空間就學會排版了！Knuth 教授在他的 *The TEXbook* 一書中也曾形容使用 TEX 排版的情形：一個版面就像一個含有膠水（glue）的頁面，然後每一個要排版的內容就是各種不同的 box，在這些 box 還沒有固定正確位置時，都是可以移動的（膠水還沒有乾），一旦排版完成，膠水就乾了，於是每個 box 的位置就固定無法再移動了，除非又從頭再來。

一個字母、一個單字、一個句子、一個段落、一個符號、一個圖形、一個表格都可能構成一個 TEX 的 box，甚至 box 中還有 box 的情形。這章想討論的，就是這個 box 如何安置他們到正確的位置，讓每個 box 之間的空間都能達到恰到好處，所以，到底是在控制 boxes 的屬性、位置，還是調整 glue 的空間，就看各位怎麼去看待了（請注意，box 不一定是可見的！在 TEX 裡頭，glue 是可以調整的。）。

我們前面所討論到的英文句點後空白的調整、italic correction、`\linespread` 及 `\parindent` 這些都是在調整 glue。通常，在 LATEX 系統裡頭，指定單位常常不會是絕對固定的，會視情形做小限度的自動微調，這是版面空間配置上的需要。

5.1 LATEX 中使用的度量單位

要精確描述和調整 LATEX 中的空間及位置，我們必需要有個標準的度量單位。以下都是在 LATEX 常會用到的單位。這裡有絕對單位及相對單位之分，除非必要，不然，一般是建議使用相對單位，原因是，他會隨著文稿字型大小改變時跟著做適當的調整。當然，在很講求精確、固定大小的顯示時，就得使用絕對單位了。

這裡如果是閱覽 HTML 格式版本，請另參考 PDF 格式版本，以免表示上失眞。以下表格中所畫出來的長度僅供參考用。

5.1.1 絕對單位

單位名稱	意義	長度
pt	point, 1/72.27 inch	⊦
bp	Adobe big point, 1/72 inch	⊦
pc	pica, 12pt	⊢⊣
mm	millimeter, 1/25.4 inch	⊦
cm	centimeter, 10mm	⊢——⊣
in	inch, 25.4mm	⊢————————⊣

這裡要注意的是 TEX/LATEX 系統中所謂的點（point），指的是一般的 printer point，也就是 1/72.27 inch，但在 Adobe 的規格中，例如 PostScript 語言中的所謂點，他是 big point，等於 1/72 inch（小數點的部份捨去了），會比一般的 print point 稍微大一點點。

5.1.2 相對單位

單位名稱	意義	長度
em	約正在使用字型字母 M 的寬度	⊢⊣
ex	約正在使用字型字母 x 的高度	⊦

在 TEX 裡頭所謂的 em，其實，精確而言是指在 Knuth 教授設計的 Computer Modern 字型裡頭的 em-dash 的寬度，由於字母 M 實際上是包在字型上所謂的 em-square 假想方格中，而 em 所指的寬度是指這個 em-square 的寬度，但字母 M 本身並不全佔有這個 em-square，因此這樣就會造成差異了。所以以字母 M 的寬度來說明的話容易有疑義。LATEX 有個指令 \quad 這就是產生一個正確 em 的寬度的空白，所以在 Knuth 教授的 *The TEXbook* 中，說明 em 就直接說他是一個 'quad' 的寬度。

5.2 版面大小

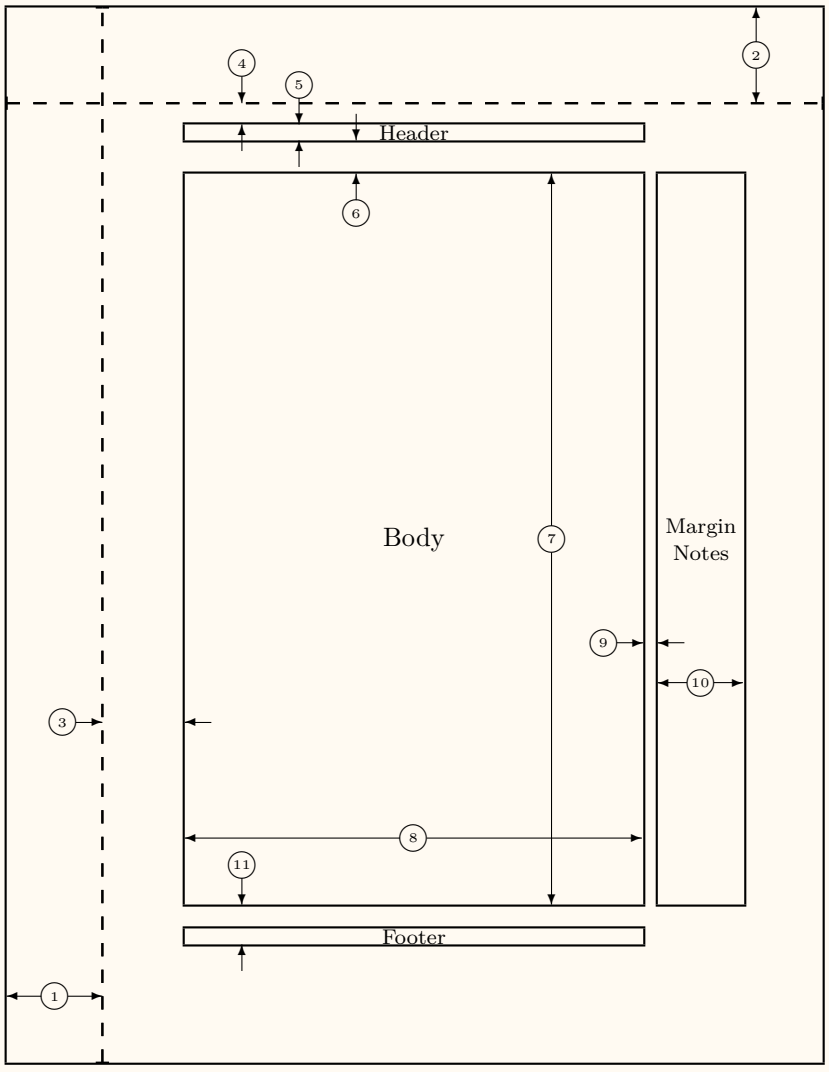
我們對於所能控制的一整張紙的範圍都可以稱爲版面。當然，我們的內文（body）並不是佔滿整張紙的範圍，上下左右都會留有一定的空白。小時候在宣紙上練習寫毛筆，老一輩的都會要我們留「天地」，這就是指內文四周的空白，除了視覺上的理由，大概也是人生的哲理吧？:-)

在編輯上，也有人稱內文（body）的部份爲「版心」或「版口」，四周的空白部份，則稱爲「版邊」。突破版心、版邊的設計，就稱之爲「出血」，例如，以背景圖佈滿整張紙當

做是背景的場合，以這個背景圖而言，就無所謂版邊了。但這在 LaTeX 通常是不會有這種情況出現，除非特意去指定內文和紙張大小同樣範圍。

當然，在內文以外的空白，也並非全是空白，他包含了頁足（footer），頁眉（header）及邊註（marginal note）的部份，記載關於頁數、註解等資訊。

5.2.1 版面圖解




```
 1   one inch + \hoffset          2   one inch + \voffset
 3   \oddsidemargin = 62pt        4   \topmargin = 16pt
 5   \headheight = 12pt           6   \headsep = 25pt
 7   \textheight = 550pt          8   \textwidth = 345pt
 9   \marginparsep = 11pt        10   \marginparwidth = 65pt
11   \footskip = 30pt                 \marginparpush = 5pt (not shown)
     \hoffset = 0pt                   \voffset = 0pt
     \paperwidth = 614pt              \paperheight = 794pt
```

這裡所謂的紙張大小，指的是 paperwidth 和 paperheight 所圍成的範圍，並非實際上手上拿到的紙張大小，實際在手上的紙張通常會略大於我們這裡的所謂紙張，所以，正式列印時，還需做微調或截切才會是眞正的這裡所謂的紙張大小（版面大小）。

這是 10pt 內文大小，如果不指定紙張的話，LaTeX 預設會使用美式 letterpaper 的大小，如要使用歐、日式的 a4paper 的話，要另行指定。我們可以稍微看一下 LaTeX 預設是如何安排版面空間的。其中 Header（頁眉）、Footer（頁足）及邊註的空間是不含括在內文 Body 裡頭的，這裡是只是單面的圖，如果是雙面的話，那偶數頁和奇數頁的邊註是要左右對換的，也就是說這個圖是奇數頁，偶數頁的話，邊註是在左邊。

這裡我們來看一下這些值所代表的意義：

指令（值）	意義
\paperwidth	紙張的寬度
\paperheight	紙張的高度
\textwidth	內文（body）的寬度
\textheight	內文（body）的高度
\headheight	頁眉（header）長度
\headsep	頁眉與內文間的距離
\footskip	內文底至頁足底之距離
\topmargin	頁眉上方的空白
\marginparwidth	邊註的寬度
\marginparsep	邊註與內文的距離
\marginparpush	兩邊註間距
\oddsidemargin	內文左邊的空白大小
\hoffset	微調版面在實際紙張的左右位置
\voffset	微調版面在實際紙張的上下位置

\hoffset 及 \voffset 就是在調整版面在實際紙張上的正確位置，這樣印出來的時候才會在實際紙張的中央。

頭昏了嗎？這很正常，因爲 LaTeX 的版面設定對初接觸的人來說，是惡名昭彰的困難、麻煩，因此這裡不多談他的設定，剛開始實在沒有必要把時間花在這個地方。如果實際想調整版面，建議使用 geometry package。舉個例子，想讓各邊緣是 2cm 就好，那只要在 preamble 區設定：

```
\usepackage[margin=2cm]{geometry}
```

就可以了，如果以 12pt 大小的字，a4paper 紙張大小的設定的話，以中文而言，大約是每行 40 個中文字，這是內文的寬度。可以視情形自行調整 margin 的值就行了。我們很

希望，下一版的 LaTeX 能對這方面做改善，以方便使用者設定。

5.2.2 紙張大小

紙張	大小	紙張	大小
a4paper	21x29.7cm	letterpaper	8.5x11in
a5paper	14.8x21cm	legalpaper	8.5x14in
b5paper	17.6x25cm	executivepaper	7.25x10.5in

至於如何指定紙張大小，這裡先簡單說明一下這篇文章的設定，談到 LaTeX 的文稿類別時會再詳細說明。

```
% 本文的設類別設定
\documentclass[12pt,a4paper]{report}
```

所以，這篇文章使用的是 a4paper，內文字型的大小是 12pt。方括號的參數是選項，可以省略，如果省略的話，預設值就是 10pt/letterpaper。

5.3 調整橫向空間

這裡的橫向空間，例如 ~ 這單字間正常空白，及 \quad 這個 em 寬度空白，都是在調整橫向的空白。但如果是要更大、或更小的空白時該如何調整呢？底下我們就來看看 LaTeX 中有什麼控制指令可以運用：

5.3.1 調整橫向空間的指令

指令	意義
\hspace{單位}	向右空出某個度量單位的空白，如果是負數，則是向左
\hfill	讓左右兩旁的文字往兩邊擴張至一個行寬爲止
\quad	空出一個 em 單位的空白
\qquad	空出二個 em 單位的空白
\thinspace	空出 1/12 個 em 單位的空白
\enspace	空出 1/2 個 em 單位的空白
\dotfill	作用和 \hfill 相同，只是空白變成點
\hrulefill	作用和 \hfill 相同，只是空白變成一橫線
\centering	此指令以後的文字將會居中排列，左右沿將不切齊
\raggedright	此指令以後的文字將會居左排列，右沿將不切齊
\raggedleft	此指令以後的文字將會居右排列，左沿將不切齊
\centerline{}	將大括號內的文字居中排列

一行的行首使用 \hspace{單位} 時將會失效，這時可以加個星號，例如 \hspace*{3em}。

在使用 \centerline{} 的場合，對於短文句很方便，因爲他不會影響以下的文字，但他也不會折行，甚至加入換行符號也無效。所以如果文句長度超過一行的行寬，他會超出邊界，甚至就首尾的文字就看不到了。

其中 \thinspace 又可以使用簡化的 \, 來代替，主要是用在引號中又有引號的情形，通常這種情形，兩引號之間要有個間隔，以便區分，例如：

```
``\,`Superman', he said.''
```

表現出來會是：

“ ‘Superman’, he said.”

這樣才能區分出不同引號，否則會變成一個連三個 grave accent 的引號。請注意，由於這個指令是由非字母符號構成，所以，他的作用範圍在遇到符號本身後就結束了，後面不必空白或以大括號來限制其作用範圍，就好像曾提到過的 \@ 一樣的情形。

5.3.2 調整橫向空間的環境

`\begin{center}...\end{center}`	讓這個環境內的內容置中
`\begin{flushleft}...\end{flushleft}`	讓這個環境內的內容靠左
`\begin{flushright}...\end{flushright}`	讓這個環境內的內容靠右
`\begin{raggedright}...\end{raggedright}`	讓這個環境內的內容靠左，右沿將不切齊
`\begin{raggedleft}...\end{raggedleft}`	讓這個環境內的內容靠右，左沿將不切齊

進入環境，和上一節提到的指令，兩者有什麼不同呢？最大的不同是，這可以方便的指定一個範圍的文句讓他作用，而不會影響環境以外的文句。其次，進入環境，縱使和上下行連在一起，沒有空出空白行，他也會自動的在上、下行空出個空白行出來，使用指令的話則不會。

咦！這裡怎麼又有個 `raggedright` 及 `raggedleft`？原來他也是可以當環境來使用。由於這兩個指令會使以下的內容的左、右沿不切齊，因此使用上要非常小心，除非本來就想讓內文的左、右沿不切齊，否則，最好是使用有範圍限制的方式。當然，如果這兩個指令是用在某個其他環境範圍內，他的作用也將僅限於這個環境內，不會影響這個環境外的文句。

5.3.3 引文環境

引文通常就是引用他人的文句，在引文的段落，兩旁都會出現內縮的情形，以便和正文相區隔，這也是一種空間的配置，可增加文章的易讀性。在 LaTeX 裡頭有三種引文環境：`quote`, `quotation`, `verse`。這三者看起來很像，但有些微的差異。

環境	適用時機	特性
`quote`	較短的短引文	每個段落第一行不內縮
`quotation`	多個段落的長引文	每個段落第一行會內縮
`verse`	詩歌、詞引文	每個段落的第一行不內縮，但第二行起會內縮

在 `verse` 的情形，通常會使用 `\\` 來換行以便控制每一行的寬度。而且段落間距將不受外在設定的影響，其中 `quote` 和 `verse` 環境會預插入適當的段落間距，而 `quotation` 環境則不會。

底下我們來看看調整橫向空間的一個綜合實例：

```
% example11.tex
\documentclass{article}
\usepackage{CJK}
```

```
\begin{document}
\begin{CJK}{Bg5}{hwmm}
\section{hspace}
\hspace*{2em}這是一個橫向空間調整的測試。\\
這是一個\hspace{2em}橫向空間調整的測試。\\
這是一個 \hspace{2em} 橫向空間調整的測試。
\section{hfill}
這是一個\hfill{}橫向空間調整的測試。
\section{quad}
這是一個\quad{}橫向空間調整的測試。\\
這是一個 \quad{} 橫向空間調整的測試。\\
這是一個\qquad{}橫向空間調整的測試。
\section{dotfill}
這是一個\dotfill{}橫向空間調整的測試。\\
這是一個 \dotfill{} 橫向空間調整的測試。
\section{hrulefill}
這是一個\hrulefill{}橫向空間調整的測試。
\section{center}
\begin{center}
這是一個橫向空間調整的測試。
\end{center}
\section{flushleft}
\begin{flushleft}
這是一個橫向空間調整的測試。
\end{flushleft}
\section{flushright}
\begin{flushright}
這是一個橫向空間調整的測試。
\end{flushright}
\section{quote}
這是節錄自伊索寓言的節錄故事：
\begin{quote}
An antwent to the bank of a river to quench its thirst, and
being carried away by the rush of the stream, was on the
point of drowning.
A Dove sitting on a tree overhanging the water plucked a
leaf and let it fall into the stream close to her. The Ant
climbed onto it and floated in safety to the bank.
\end{quote}
\section{quotation}
這是節錄自伊索寓言的節錄故事：
\begin{quotation}
An antwent to the bank of a river to quench its thirst, and
being carried away by the rush of the stream, was on the
point of drowning.
A Dove sitting on a tree overhanging the water plucked a
leaf and let it fall into the stream close to her. The Ant
climbed onto it and floated in safety to the bank.
```

```
\end{quotation}
\section{verse}
這是節錄自伊索寓言的節錄故事，這是節錄自伊索寓言的節錄故事，%
這是節錄自伊索寓言的節錄故事，這是節錄自伊索寓言的節錄故事：
\begin{verse}
An antwent to the bank of a river to quench its thirst, and
being carried away by the rush of the stream, was on the
point of drowning.
A Dove sitting on a tree overhanging the water plucked a
leaf and let it fall into the stream close to her. The Ant
climbed onto it and floated in safety to the bank.
\end{verse}
\section{centering}
\centering
這是一個橫向空間調整的測試。\\ % 這裡要換行，否則會是 \raggedright 的作用
\raggedright
\section{centerline}
\centerline{這是一個橫向空間調整的測試。}
\section{raggedright}
\raggedright
這是一個橫向空間調整的測試。
\section{raggedleft}
\raggedleft
這是一個橫向空間調整的測試。
\end{CJK}
\end{document}
```

編譯好的結果如下：

http://edt1023.sayya.org/tex/latex123/example11.tex
http://edt1023.sayya.org/tex/latex123/example11.pdf

要注意是指令前後的空白，像 \hspace, \dotfill, \hrulell 這類指令，指令前後空白都會算進去的。\quad, \qquad 這類指令，則後面的空白也是會算入的。另外，由例子中可以看出來，一個 em 的寬度，大約是一個中文字的寬度，所以，我們預設使用 10pt 的字，這個 em 寬度就相當於 10pt 的寬度，所以，我們在第一行插入了 2em 寬度的空白，也就好像是內縮了兩個中文字一樣。

5.4 調整縱向空間

\vspace{單位}	向下空出某個單位的空白（行），負數則是向上
\bigskip	產生 12pt（11–12pt）的垂直空白（行）
\medskip	產生 6pt（5–7pt）的垂直空白（行）
\smallskip	產生 3pt（2–4pt）的垂直空白（行）
\vfill	和 \hfill 類似，作用是將某段落向上頂，或往下擠
\parskip 單位 =	調整全文每個段落間的距離爲某個單位

其中的 \bigskip, \medskip, \smallskip 並非固定的，他們會視上下文脈絡的需要自動做微調，以達到一整頁較一致的空間配置。\vspace 如果是出現在一頁的第一行或最後一行時，將會失去作用，這時可以加個星號，\vspace*{單位}。

爲了維持版面的一致性，使用縱向空間調整的指令時要特別留意，例如章節標題上下的空間、各段落間的空間，進入環境前後所空出的空間，這都有一個固定值，LaTeX 會自動去調整，不必由使用者自行動手，除非是封面這種單獨頁。所以，使用縱向空間調整指令時，要非常注意整體的一致性，這也是排版上的一個很重要的原則。

這裡舉這篇文章的內頁封面爲例來綜合說明，橫、縱向空間的運用。還記得第 4.3 節的 title page 的指令嗎？其實我們也可以自行設計一個獨立的內頁封面，使用的是 LaTeX 本身的 titlepage 環境。這裡的圖檔引用是我們還沒有學習到的，沒關係，只要大原則抓住就行了。

```
% example12.tex
\documentclass[12pt,a4paper]{report}
\usepackage{CJK}      % 引入所需要的 packages
\usepackage{graphicx}
\begin{document}
\begin{CJK}{Bg5}{hwmm}
\begin{titlepage}     % 使用 titlepage 環境
\vspace*{5ex}
  \begin{flushright} % 大標題靠右
    \Huge\textbf{大家來學 \LaTeX}
  \end{flushright}
  \rule{\textwidth}{.256ex}
  \begin{flushleft}  % 版本號碼及日期靠左，和大標題之間以一橫線隔開
    Version 0.1 draft\\
    \today
  \end{flushleft}     % 圖檔位於中央偏左
  \vspace{8ex}        % 空出 8ex 的垂直空間
  \hspace{2em}\includegraphics[scale=.75]{cover2.1} % 引入圖檔，並將這個
  \vspace{8ex}                                      % 圖檔橫向右移 2em
  \begin{flushright} % 作者資訊靠右
```

```
      By Edward G.J. Lee 李果正\\
      Email : \texttt{edt1023@info.sayya.org}
    \end{flushright}
  \end{titlepage}
  \end{CJK}
  \end{document}
```

由於配合版面的問題，其中有一些數據有更動，而且也省略了一些我們還沒有學習到的 packages，但大結構則和原始文稿一樣。所以，和這篇文章的 PDF 格式比較會發現，大標題的字體小了一點，而且沒有顏色，也沒有超連結。

使用 titlepage 環境後，在 report/book 文稿他會自成一沒有頁數的單獨頁，在 article 類別，因爲會和內文連接，所以，在選項的部份要多加一個 titlepage 的選項。另外，在 titlepage 裡頭就不能再使用 \title, \author 指令了，在本文的地方也不必再下 \maketitle 指令。

編譯好的例子如下：

http://edt1023.sayya.org/tex/latex123/example12.tex
http://edt1023.sayya.org/tex/latex123/example12.pdf

這裡要特別說明的是，引入圖檔要使用 graphicx package（這是最常用的，也有其他的方法來引用），引入的指令是 \includegraphics，這個我們會在第 9 章會討論，在這裡我們把圖檔縮小成爲原圖的 75%，否則整個封面會超出一頁。這個圖檔是由 METAPOST 檔案所編譯而來的，他是一種 eps 圖檔（簡單的說，是含有邊界數據去除不必要周圍空白的 ps 檔，方便引入或輸出至文稿中），編譯的方法如下：

```
mpost cover2.mp
```

這樣就會產生 cover2.1 這個 eps 圖檔，這樣就可以直接引入了。他的原始碼及 eps 圖檔在：

http://edt1023.sayya.org/tex/latex123/cover2.mp
http://edt1023.sayya.org/tex/latex123/cover2.1

5.5 條列環境

條列環境也是屬於一種空間的控制，他把一些文字按一定的方式來排列，條列環境中一些起頭的符號、文數字或字串，我們稱之爲項目標籤（item label），利用這些不一樣的排列位置及不一樣的項目標籤起頭來敘述文句，就可以達到醒目的作用。這是以章節分隔以

外，相當常用讓內容一目了然的方法，建議多多利用。請千萬記得，環境中還可以有環境，而且以下三種的條列方式可以混合交叉使用。

5.5.1 項目式條列環境（itemize）

這是以符號來起頭醒目的一種條列方式。例如：

```
\begin{itemize}
\item 第一大項，這裡是第一大項。
\item 第二大項，這裡是第二大項。
 \begin{itemize}
 \item 第一小項，這裡是第一小項。
 \item 第二小項，這裡是第二小項。
 \end{itemize}
\item 第三大項，這裡是第三大項。
\item 第四大項，這裡是第四大項。
\end{itemize}
```

排版出來會變成：

- 第一大項，這裡是第一大項。
- 第二大項，這裡是第二大項。
  - 第一小項，這裡是第一小項。
  - 第二小項，這裡是第二小項。
- 第三大項，這裡是第三大項。
- 第四大項，這裡是第四大項。

5.5.2 列舉式條列環境（enumerate）

這是以數目字或字母或羅馬數字來起頭醒目的條列方式。同樣的例子，改成 `enumerate` 的話，會排版成：

1. 第一大項，這裡是第一大項。
2. 第二大項，這裡是第二大項。
   (a) 第一小項，這裡是第一小項。
   (b) 第二小項，這裡是第二小項。
3. 第三大項，這裡是第三大項。
4. 第四大項，這裡是第四大項。

5.5.3 敘述式條列環境（description）

這是以一個簡短文字敘述來起頭醒目的條列方式。再把他改成 description 的話，他是以粗體文字來起頭醒目的，這些文字要用方括號括住。

```
\begin{description}
\item[第一大項] 這裡是第一大項。
\item[第二大項] 這裡是第二大項。
  \begin{description}
  \item[第一小項] 這裡是第一小項。
  \item[第二小項] 這裡是第二小項。
  \end{description}
\item[第三大項] 這裡是第三大項。
\item[第四大項] 這裡是第四大項。
\end{description}
```

排版出來的結果是：

第一大項 這裡是第一大項。

第二大項 這裡是第二大項。

第一小項 這裡是第一小項。

第二小項 這裡是第二小項。

第三大項 這裡是第三大項。

第四大項 這裡是第四大項。

要注意的是，不管哪一種的條列環境，每個項目（item）的文字敘述會自動折行，這相當方便，使用者只要把條列的結構弄妥，專心打每個項目的內容就成了。而且，如果使用方括號括住一些字元、字串或符號，那帶頭的標示將會是這些字元、字串或符號，如果是列舉式的條列方式，那麼有方括號的將不被編號，會自動跳過，編號順序則會自動順延。

5.6 線框

線框在排版上佔有一定的地位，因爲他可以區隔不同的空間，也可以讓某些部份突顯出來。但是，過多的線框也是會有喧賓奪主的不好副作用，使用上可能要適可而止。這裡我們要談的是單純的線框，表格也是線框的一種應用，我們將會在第 9 章，再來談表格的處理。

5.6.1 直線（rule）

直線也是屬於方框的一種，就是一個實體長方形，只不過，他的高或寬（線的組細）只有一點點，所以，看起來就一像直線罷了。

```
\rule[上下位置(單位)]{寬}{高}
```

寬及高應不必多做解釋，那個上下位置是什麼呢？這個上下位置是和基線（baseline）在比較的，如果沒有指定，那就是在基線的位置，如果有指定，就依正負值調整和基線的相對位置，正值由基線向上調整，負值則由基線向下調整。這裡來看個個實例就瞭解了：

```
% example13.tex
\documentclass{article}
\parskip=3pt
\parindent=0pt
\begin{document}
This is a line.       % 畫高 1pt 寬 3cm 的橫線。
\rule{3cm}{1pt}
\rule[1ex]{3cm}{1pt}
\rule[-1ex]{3cm}{1pt}
\rule{1pt}{3cm}      % 畫高 3cm 寬 1pt 的直線。
\rule{3cm}{0pt}TEST. % 把 TEST 向右推 3cm。
\rule{2cm}{3cm}      % 畫高 3cm 寬 2cm 的實體方框。
\textcolor{blue}{This is color lines.}
\textcolor{red}{\rule{3cm}{1pt}}        % 有顏色的線框。
\textcolor{green}{\rule[1ex]{3cm}{1pt}}
\textcolor{blue}{\rule[-1ex]{3cm}{1pt}}
\end{document}
```

編譯好的例子在：

http://edt1023.sayya.org/tex/latex123/example13.tex
http://edt1023.sayya.org/tex/latex123/example13.pdf

由例子中可以看得出來，這個畫線指令並不是單單縱、橫畫線而已，靈活運用的話，可以做出許多不同的效果，例如方條圖之類的。

5.6.2 文字底線（underline）

有時候我們希望在書寫文字的同時，也在其下畫線。LaTeX 有現成的指令可以使用，那就是 \underline{文字}，會在文字底線的部份同時畫上線條。當然，使用上常常會和現在的橫線搞混，因此使用時要特別留意整個頁面上是否已經存在了許多橫線。

5.6.3 方框（box）

這裡主要指的文字方框，繪圖上的一般框形在第 9 章時再來討論。這一章的開頭就已談到，整篇文章是由一個個的 box 所構成的，這裡的方框則是一個典型的 box，他的地位，當然是和前面所說的 box 一樣，TEX/LATEX 會把他視爲一個單一的字母來處理。

我們先來看看最簡單的方框的指令：

指令	意義及作用
`\frame{文字}`	將文字內容以可見的方框框住，方框和文字間沒有間距
`\fbox{文字}`	將文字內容以可見的方框框住，方框和文字間有一定間距
`\mbox{文字}`	作用和 `\fbox` 同，但方框不可見

以上的指令都可以形成方框文字。

方框也有可以調整的指令：

指令	意義及作用
`\framebox[寬度][對齊方式]{文字}`	與 `\fbox` 同，但可指定寬度及對齊方式
`\makebox[寬度][對齊方式]{文字}`	與 `\mbox` 同，但可指定寬度及對齊方式

這裡的寬度指的是方框的寬度，不指定的話，作用就如同 `\fbox` 及 `\mbox` 一樣，以包住整個文字範圍爲寬度，由於我們不容易測知文字內容的寬度有多少，因此寬度的指定可以使用下列相對單位的方式：

寬度	作用
`\width`	這個寬度就是 `\fbox` 圍住時的方框寬度
`\height`	這是正常基線至框頂的高度
`\depth`	這是正常基線至框底的高度
`\totalheight`	這是 `\height` 和 `\depth` 之總和

例如，我們指定 `2\width` 他的意思就是以 `\fbox` 包住此文字內容時，其方框寬度的二倍寬。參數裡頭的對齊方式，可有下列幾種：

對齊方式	作用
`c`	center，文字位於方框中央，這是預設值
`l`	flushleft，文字位於方框左方
`r`	ushright，文字位於方框右方
`s`	stretch，文字平均分分布於方框中

不指定的話，當然就是位於方框中央位置了。我們也可以指定可見方框的線的粗細，及方框和文字間的間隔距離：

指令	作用
\fboxrule=單位	指定方框線條粗細
\fboxsep=單位	指定文字和框緣的間距

請注意，這不會影響 \frame{}。如果想特別的置放方框的位置，也可以使用 \raisebox 指令：

```
\raisebox{上下位置(單位)}[深度][高度]{文字內容}
```

這裡的上下位置的意義和 \rule 指令裡頭的一樣，只不過，這裡的上下位置一定要指定，沒有預設值，不指定會編譯錯誤。請千萬記得，方框中當然是還可以有方框的。我們現在就來看一個綜合的實例：

```
% example14.tex
\documentclass{article}
\parskip=3ex
\parindent=0pt
\begin{document}
\frame{This is frame.}
\mbox{This is mbox.}
\fbox{This is fbox.}
\framebox{This is a framebox with no argumant.}
\framebox[1.5\width]{This is a framebox.}
\framebox[1.5\width][l]{This is a framebox with \texttt{l}.}
\framebox[1.5\width][r]{This is a framebox with \texttt{r}.}
\framebox[1.5\width][s]{This is a framebox with \texttt{s}.}
This is baseline.
\raisebox{3ex}[5\height]{This is a raisebox which lift 3ex.}
This is baseline.
\fbox{\raisebox{-3ex}[5\height]{This is a raisebox which lift $-$3ex.}}
\fboxrule=1.5pt
\fboxsep=8pt
\framebox[1.5\width][s]{This is a framebox with \texttt{s}.}
\end{document}
```

這應無需多加解釋，編譯好的例子在：

http://edt1023.sayya.org/tex/latex123/example14.tex
http://edt1023.sayya.org/tex/latex123/example14.pdf

5.6.4 段落方框

另外，也有用於段落文字的方框，可以控制某個段落出現在某特定的空間、位置。我們常常把這種安排稱爲迷你版面，把版面內容置於一個不可見的方框當中，當然，這個方框和

一般的 box，例如字母，仍然處於同樣的地位，LaTeX 會把他當成一個字母單位來處理。

```
\parbox[對齊方式][高度][內文位置]{寬度}{文字內容}
\begin{minipage}[對齊方式][高度][內文位置]{寬度}
  段落內容
\end{minipage}
```

這裡的第一個選擇性參數「對齊方式」:

- `t` top，段落方框的上沿對齊一行的基線
- `b` bottom，段落方框的下沿對齊一行的基線
- `c` center，段落方框的中央對齊一行的基線，這是預設

`\parbox` 指令通常用於較短的文字內容，如果是較長的段落，那使用 `minipage` 環境會比較方便。這些段落方框和上面所談到的一些方框最大的不同是，段落方框本就是用來排版小段落文句，因此他會和一般正常文章段落一樣的處理，例如他會自動斷行，碰到空白行也會起新段落，但各段落預設是不縮排的，而且，在安排上會比一般的段落緊湊，一些邊緣的間距會縮減成 0pt。

高度不去指定的話，那就是以整個版面經斷行處理後，整個文字段落所形成的高度。至於內文位置指的也是 `t`, `b`, `c` 這些，意思是文字段落在方框內的上下位置，當然，這要有指定方框高度時才會有意義，因此，「內文位置」和「高度」是要同時指定的，要非常注意的是，如果指定了高度，但沒有指定內文位置，則預設是等於前面使用的「對齊方式」所指定的參數。

這些參數的使用會有些煩複，但這是彈性所帶來的「必要之惡」，不必去死記，只有在第一次接觸時把他搞清楚就夠了，實際要用到時再來查他的詳細參數，常用的指令、環境，大概多查幾次就自然而然的記起來了。

第 6 章

LaTeX 的標準文稿類別

這章主要是在說明 LaTeX 文稿的類別（document class）[1]，這是 LaTeX 規範文稿整體結構的方法。使用 class 的用意，就是把版面結構處理和實際文稿分開，這樣的最大好處就是維持整篇文章排版結構上的一致性，也使文稿內容更清爽簡潔，使用者只要專心於文稿內容的寫作即可，如果 class 定義的好，也可以達到一文多變化又不變更文稿內容的目的，只要把引用的 class 換成別的就可以了，其他的可以不必更動。

目前，LaTeX 有五種標準類別用於一般文件，可用於一般的書信、雜誌、期刊、報告及論文。但有些期刊、論文會要求一定的結構，這時得依需求另行訂定。因此，也有其他的類別存在，標準類別並不是唯一的。甚至，也可以自行撰寫自己的文稿類別。當然，我們一般使用是不需要這麼講究，這裡只介紹 LaTeX 的標準類別。而且，如果有和他人交換文稿的需求時，我們應該盡可能的使用流通性較廣泛的類別。

另外，也有一些是關於 LaTeX 說明文件及 macro 寫作時要用到的類別，這些已超出本篇文章的範圍。

6.1 LaTeX 類別的宣告

LaTeX 的類別，要在文稿的一開頭時就宣告（當然，其上有註解是沒有關係的），他的一般格式如下：

```
\documentclass[選擇性參數]{類別}
```

選擇性參數是可以省略的，但類別名稱則不能省，一定要指定一個類別。而且只能只有一個類別。

[1]這在舊的版本稱爲 style，這兩個詞意義上差不多，這些都是 TeX 延伸出來的巨集定義，專門用來定義文章的大結構，以便簡化使用上及文稿的內容。

6.2 類別的選擇性參數

選擇性參數可以選擇多個，各個選項是以逗點分開的。

1. `10pt, 11pt, 12pt`
   指定內文一般正常字的大小，預設是 `10pt`。其他點數沒有外來 package 的幫忙不能指定。

2. `a4paper, letterpaper, b5paper, executivepaper, legalpaper`
   指定紙張大小，預設是 `letterpaper`。

3. `fleqn`
   使數學式靠左對齊，預設是居中對齊。

4. `leqno`
   使數學式編號靠左，預設是靠右。

5. `titlepage, notitlepage`
   決定 title page 是否獨佔一頁。預設 `article` 不獨佔一頁，但 `report/book` 則會獨佔一頁，在這裡是可以指定來變更預設行爲，例如 `article` 文稿，指定 `titlepage` 的話，那 title page 就會獨佔一頁。

6. `onecolumn, twocolumn`
   文章單欄或兩欄式，預設是單欄，也就是不分欄。

7. `twoside, oneside`
   是否區分奇偶數頁。預設 `article/report` 不區分，`book` 則會區分。一般的書籍，在裝訂的部份，他的中央線稱爲書脊，偶數頁會在打開書籍時的左方，而且其中內容會偏向書脊的中央（此時是向右），反之，奇數頁會在右，內容一樣會偏左向書脊，在 `oneside` 的情形則不做這樣的區分，不管奇偶頁都會在紙張的中央部位。

8. `landscape`
   橫向列印或縱向列印，預設縱向（portait）。

9. `draft`
   草稿式編譯，這時圖檔將不會被引入，可加快編譯的速度。不過，如果編譯是使用向量字型的話，編譯速度應該是還算很快。但使用 `draft` 的一個好處是，過長的地方會標示出來。

10. `openright, openany`
    這是在控制，章的開始是否是奇數頁（right-hand page）。在 `book` 類別，預設章會從奇數頁開始，report 類別則不會。`article` 類別沒有章，所以，對此一設定會忽略。

6.3 類別的種類

這裡只列出一般文章使用的類別，其他特殊巨集或 LaTeX 正式文件所使用的類別就不列出了，一般使用，這些類別就足夠了。

類別	一般用途	特性
`article`	一般短文	無章，連續頁方式的安排，無奇偶數頁的區分
`report`	較長論文	章會起新頁，預設無奇偶數頁的區分
`book`	書籍類	章會於奇數頁起新頁，預設有偶數頁的區分
`letter`	信件	英文信件格式
`slides`	幻燈片	幾乎另用外來套件取代
`minimal`	測試及寫新類別	這是最簡單的類別，只規定了內文的寬、高，正常字

當然，這些用途並不是固定不變的，得看使用者的安排，不想多花時間、精神的話，那就依 LaTeX 預設的格式去使用，至少就不會太離譜。其中 `minimal` class 是用來測試用的，或者寫新的 class 用的，他完全沒有版面的安排，例如，沒有章節的結構，各種間距也沒有定義，預設的字型是正常字，沒有其他的變化，幾乎所有的變化要自行去定義。

第 7 章

巨集套件

LaTeX 系統已經好久沒有更新，有些部份可能會跟不上實際的腳步，而且有些內定的巨集定義，經過大家的使用，發覺並不是那麼的順手，尤其是功能的強化方面，因此這章談談如何引用他人已經寫好的巨集，這很重要，盡量避免重複製造輪子，寫 TeX/LaTeX macro 可說是很專業的工作，要避免破壞了整體的結構，所以先找看看有什麼巨集套件可以使用。

7.1 一般套件的使用

我們曾在第 3.4.3 小節，頁 22，提到過簡單巨集的引用，事實上，有些巨集含有許多的參數來做微調，但是每個巨集套件的參數都不會一樣，因此，使用套件之前要先看一看他所附上的使用手冊。幾乎大部份的巨集套件都有使用手冊，如果是系統上就有的巨集，那麼這些文件通常會放在：

```
$TEXMF/doc  => Unix-like 系統
$TEXMF\doc  => DOS/Windows 系統
```

這些目錄底下，這些文件會有原始 TeX/LaTeX 文稿，也有編譯好的 `*.dvi` 或 PostScript 檔可以閱覽，爲求方便的話，可以將他們轉成 pdf 格式來閱覽，原因是可以以關鍵字來搜尋全文，在查指令、環境時會比較方便。在 Unix-like 系統或 Windows 下的 cygwin 環境的話，可以使用 `texdoc` 這個指令來閱覽，例如：

```
texdoc amsguide  => 閱覽 amsguide.dvi 這個檔的說明
texdoc -s ams    => 查系統上所有含 ams 字樣的文件
```

7.2 LaTeX 官方文件中的標準巨集套件

底下是 LaTeX 官方文件中所附的標準巨集套件。雖然是標準巨集套件，但一般情形下，使用這些 packages 的機會並不多，都是有特殊需要時才會引入。

7.2.1 alltt

這個巨集套件提供 `alltt` 環境，和 `verbatim` 環境的作用相同，只是 \，{，} 的作用和一般文章中相同會被 LaTeX 解讀。這有什麼用呢？這樣一來 LaTeX 這個特殊標誌也可以使用，也可以讓環境中的文字具有顏色，或做其他變化，當然，裡頭的文字預設仍然是使用打字機字族的。

7.2.2 doc

這是用來寫 LaTeX 文件的巨集套件，這在使用 `ltxdoc` 這個 class 的同時就會載入 doc package。由於這不是用於一般的文件使用場合，所以，這裡就不多談，有興趣的話可自行參考他的文件說明。

7.2.3 exscale

由於原先的 Computer Modern font 中的數學延伸符號（`cmex`）只有 10pt 大小的字型（`cmex10`），內文放大到 `large` 以上的字型時，例如放大到 `Large` 時，有些數學符號仍然會維持一定的大小，這時可以使用這個套件，讓這些數學符號也跟著放大，例如積分符號。exscale 只有字型縮放的定義，因此只要把這個套件在 preamble 區引用就可以了，無需任何指令。

不過，這裡要說明一下，在 TeX/LaTeX 裡字型放大，有時可能會造成表現失眞的情形，尤其是數學式子，爲了顧及數學式子中各個字母間的空間安排，`cmex10` 的設計並不適合拿來放大，可以把 `cmr5` 放大成 10pt 和眞正的 `cmr10` 來比較就會知道表現出來會不一樣，因此，如果考慮精確配合的問題，放大數學式子的字型時可能要考慮一下使用場合，尤其目前採用向量字更是如此。請試試以下的例子，我們把 `cmr5`、`cmr10` 及 `cmr12` 同樣放大到 30pt 來看看結果會不會一樣：

```
% test-fonts.tex
```

```
\font\largecmr=cmr12 at 30pt
\largecmr
This is cmr12 at 30pt.
\font\largecmr=cmr10 at 30pt
\largecmr
This is cmr10 at 30pt.
\font\largecmr=cmr5 at 30pt
\largecmr
This is cmr5 at 30pt.
\bye
```

請注意，這是 TeX 文稿，不是 LaTeX 文稿，所以要使用 `tex` 或 `pdftex` 來編譯，他的結果如下，大家可很清楚的看得出來，雖然同樣是向量字，但放大時的表現並不會一樣：

http://edt1023.sayya.org/tex/latex123/test-fonts.tex
http://edt1023.sayya.org/tex/latex123/test-fonts.pdf

根據 Knuth 教授當初設計 METAFONT，他的理念是一個同樣的字型在放大的時候，同一個字，他的筆劃置放的相對位置應該要隨放大的倍數而稍加調整。因此，假如我們只使用一種向量字型，用在不同的放大倍率的時候，文字符號間的空間配合會產生不一樣的結果，尤其是用在數學式子的時候，更加明顯。

當然，實用上 METAFONT 雖然也是一種向量字型，但由於太過於複雜，不適合拿來螢幕顯示上用，所以才會退而求其次，轉成 pk 點陣字型來使用。這也就是爲什麼同樣是 `cmr` 的字型，會有幾種不同點數的獨立字型的原因，縱使是向量字也是如此。TeX 已經 20 幾歲了，但是，我們的字型技術似乎還是沒有完全趕上當初 Knuth 教授的理念。

7.2.4 fontenc

在第 4.7.1 小節，頁 32，曾提到字型編碼的問題。要改變字型編碼，可以使用這個 fontenc package。以 T1 font encoding 來說：

```
  ...
\usepackage[T1]{fontenc}
  ...
```

這樣就可以了，但由於一些字型，例如歐洲字元，在原來的 Computer modern Type1 字型中安排不一樣，所以，有些部份會使用原始的 METAFONT 字型所轉換成的 pk 點陣字，這樣的話，一般印表機印出來是差異不大，但如果是想製作成 PDF 格式在螢光幕閱覽的話，字型的表現會變得很醜。理想的話，要安裝 `cm-super` Type1 字型，但是一般使用者

恐怕自行安裝字型會有困難。這在 teTeX 2.x 以後的版本，已經有附上 pxfonts 及 txfonts package 及其字型，所以，如果是新近版本的 teTeX 的話，可以由以下的方式來使用：

```
  ...
\usepackage{txfonts}
\usepackage[T1]{fontenc}
  ...
```

其中 txfonts 是模擬 Times 系列的字型，pxfonts 是模擬 Palatino 系列的字型。當然，這裡關於字型的問題有點複雜，這不在這篇文章的討論範圍，只能做簡單的說明，如果沒有特殊需要，例如，歐洲字元、一些有重音符號的字母，那使用預設的 OT1 編碼就行了，因爲這些套件所附的有些字型，只有一種大小的 Type1 字型在縮放，因此使用上恐怕會有失眞的情形。

7.2.5 graphpap

這是產生方格紙的巨集。他提供了一個指令，可以畫方格，可以配合 picture 環境來使用，他的語法是：

```
  ...
\usepackage{graphpap}
  ...
\graphpaper[n](x,y)(x1,y1)
  ...
```

其中的 n 如果省略的話，預設是 10，他指的是方格紙的最小刻度單位。(x,y) 及 (x1,y1) 指的是左下角及右上角的座標值，例如：

```
\documentclass{article}
\usepackage{graphpap}
\begin{document}
\graphpaper(0,0)(360,360)
\end{document}
```

這樣會畫出以 10 爲最小刻度的方格，編譯好的例子如下：

http://edt1023.sayya.org/tex/latex123/test-graphpap.tex
http://edt1023.sayya.org/tex/latex123/test-graphpap.pdf

7.2.6 ifthen

TeX 本身是一種排版程式語言，當然會有條件判斷式來方便寫巨集，但如果文稿中也充滿了條件判斷式，將會使文稿複雜化，難以閱讀、維護，因此，一般條件判斷式大多數使用在巨集定義，而不是寫在文稿當中。這個 package 就是在簡化條件判斷式，以便也可以方便使用在文稿當中。

ifthen package 提供了 `\ifthenelse` 指令來做條件判斷。他後面有三個參數，第一個是條件式，第二個是條件爲眞的時候要執行的內容，第三個是條件爲僞的時候要執行的內容。這裡不多談他的使用，底下只提供一個實例片段：

```
  ...
\usepackage{ifthen}
  ...
\ifthenelse{\isodd{\thepage}}%
  {\setlength{\leftmargin}{10pt}}%
  {\setlength{\leftmargin}{0pt}}
  ...
```

這樣奇數頁時，`leftmargin` 會設爲 10pt，偶數頁時則爲 0pt。後面加 % 代表，這三行是一整行，其間沒有空白。

7.2.7 inputenc

由於 fontenc package 的一些字型編碼安排，和一般所謂的 Latin-1 這些編碼（input encoding），他們的內容不一定相符，所以，fontenc package 常會和 inputenc package 互相配合使用，以確保在使用歐洲字元、符號時能正確取得到字。例如：

```
  ...
\usrpackage[T1]{fontenc}
\usepackage[latin1]{inputenc}
  ...
\inputencoding{ascii} % 也可以在文稿內文變換
  ...
\inputencoding{latin2}
  ...
\inputencoding{latin1}
  ...
```

當然，我們的文稿如果只是英美語系的文章，那這些都可以不必理會。

7.2.8 latexsym

這是 LaTeX 額外提供的符號。在新版的 LaTeX 2$_\varepsilon$ 並不會自動載入，要自行引入這個獨立出來的 package。這主要是提供 `lasy*` 這些字型裡頭的符號。如果有使用 amsfonts 或 amssymb package 的話，這些 latexsym 符號應該是可以無需引入（有少數符號是 LaTeX 特有的）。至於各種符號的 package 有哪些內容，可以參考系統上的 `symbols*` 這些檔案，他可能存在的形式是：

```
symbols.dvi
symbols-a4.ps[pdf]
symbols-letter.ps[pdf]
```

或者，也可以從 CTAN 下載最新的版本：

ftp://cam.ctan.org/tex-archive/info/symbols/comprehensive/symbols-a4.pdf

7.2.9 makeidx

這是在製作索引時要引入的 package，我們會在第 11.3 節，頁 141 再來討論。

7.2.10 newlfont

這是模擬舊版 LaTeX 的字型用法，讓他使用新的取字機制的 package。也就是我們在第 4.7.2，頁 34 所提到的用法。爲免麻煩，我們盡量避免使用舊用法，而使用字型的標準指令。

7.2.11 oldlfont

這是模擬舊版 LaTeX 的字型用法的 package。

7.2.12 showidx

這個 package 會顯示，`\index` 指令下在什麼地方。這也會在第 11.3 節來討論。

7.2.13 syntonly

syntonly package 提供 `\syntaxonly` 指令，他可以檢查語法是否正確，並不會有 `*.dvi` 檔的輸出。但這個 `\syntaxonly` 指令一定要放在 preamble 區。

7.2.14 tracefnt

這是追蹤字型使用情形的 package。通常編譯時所產生的資訊已經很足夠，但如果希望有更詳細的字型使用資訊的話，可以使用這個 package：

```
  ...
\usepackage[debugshow]{tracefnt}
  ...
```

請注意，這樣會增加編譯的時間，而且 `*.log` 檔會很大。

7.3 LaTeX 官方文件中的工具組

這些巨集套件，LaTeX 官方文件是歸類在相關軟體（relative software）中，可能會比上一節提到的標準巨集套件來得實用些。但也同時可以看得出來 LaTeX 非內建的套件不少，加上其他外來的巨集套件，那眞的是套件滿天飛，我們很希望在可能的情形下 LaTeX team 可以考慮將一些必要的套件納入內建，更加落實版面處理和文稿寫作分開的理念。

7.3.1 $\mathcal{AMS}$-LaTeX

LaTeX 本身就有排版數學式子的能力，但在比較專業使用時，可能會需要增強他的功能，$\mathcal{AMS}$-LaTeX 是美國數學協會（American Mathematical Society, AMS）所發展的一個增強 LaTeX 數學式子編輯的巨集組，是由 $\mathcal{AMS}$-TeX indexamstex@$\mathcal{AMS}$-TeX 移植過來給 LaTeX 使用的，他主要分成兩個部份：amscls 及 amsmath，前者提供符合 AMS 的文件規格的文稿類別，後者可加強原來 LaTeX 的數學模式。我們會在第 10 章，頁 124 加以介紹。

7.3.2 babel

如果想排版英文以外的其他歐洲國家的語文，例如：德文、法文，那可以利用 babel 巨集套件。

7.3.3 cyrillic

這是專爲排版斯拉夫民族語文，例如：俄文，那可以使用這個套件。

7.3.4 graphics

這是處理圖形要用到的巨集套件。但目前一般都使用功能較完善的 graphicx 巨集套件來取代 graphics 了，事實上，引用 graphicx 會自動的引用 graphics，而在指令使用的方便性上，graphicx 較佳，因此我們往後都是以 graphicx 爲主來說明的。這兩個套件屬於 LaTeX 的圖形工具組，這個工具組包括了和顏色、圖形相關的各種巨集，我們會在第 9 章，頁 92 來討論。

7.3.5 psnfss

這是 Type1 字型的巨集套件組，例如：times, charter, mathptmx 等等，他會去使用這些 Type1 字型。但通常這些字型有許多是商業字型，系統上不一定會有，如果沒有的話，會去使用 free 的代替字型，或者就不嵌入這些字型了。如果沒有這些商業字型，又想要嵌入替代的 Type1 字型的話，可以考慮使用 txfonts 或 pxfonts 巨集套件及其所附字型。當然，如果專業使用的話，可能得考慮購買專業的商業字型來使用。

7.3.6 array

這是加強原來的 `array`, `tabular` 環境的巨集套件，可增許多細部微調的功能。這在第 8.4 節，頁 82，時會討論到。

7.3.7 calc

這個套件可以讓 LaTeX 接受一些簡單的代數運算。主要用於微調一些原始預設的長度及計數器（counter）。

7.3.8 dcolumn

這是讓表格中具有小數點的數字對齊的巨集套件。我們會在第 8.9 節，頁 87 中詳細討論。

7.3.9 delarray

這是加強 array 巨集套件的功能，讓矩陣或行列式的大分界符號可以使用較簡單的指令。這個套件要配合 array 巨集套件來使用。通常在 array 巨集套件中，這些矩陣或行列式的大分界符號是由 `\left` 及 `\right` 來引導才會出來，但使用 delarray 巨集則不必如此麻煩。這在第 10 章會討論到。

7.3.10 hhline

這個巨集套件會方便在畫橫線時也可以插入表格的縱線。

7.3.11 longtable

longtable 是用在跨頁表格。通常在 LaTeX 中的 `tabular` 表格是當做一個 box 來處理，因此無法再分割，所以無法跨頁來表現。這也會在第 8.10，頁 88 談到表格時提及。

7.3.12 tabularx

這是 `tabular` 表格環境的加強版，他可以方便的排版指定寬度的表格。同樣的，這會在第 8.3 節，頁 78 時提及。

7.3.13 afterpage

這個件主要在調整 LaTeX 的浮動環境（floating environment）時，置放浮動物件，例如：圖、表的位置。

7.3.14 bm

bm 的意思，就是 bold math(symbol)，這會讓數學式子以粗體的方式來顯示。這個巨集套件，提供一個 \bm{} 指令，只要把數學式子置於大括號中就會由粗體來顯示。

7.3.15 enumerate

這是加強 enumerate 列舉式條列環境的巨集套件。他可以很方便的指定要使用什麼方式來起頭，原始的 enumerate 環境，預設第一層是阿拉伯數目字，雖然也可變更，但要重新定義，不是很方便。這裡舉個例子：

```
% example15.tex
\documentclass{article}
\usepackage{enumerate}
\begin{document}
\begin{enumerate}[Example-1.]
\item This is a item 1.
\item This is a item 2.
  \begin{enumerate}[(1)]
  \item This is a item (1).
  \item This is a item (2).
  \end{enumerate}
\item This is a item 3.
\item This is a item 4.
\end{enumerate}
\end{document}
```

可以指定會順延顯示的有：A, a, I, i, 1，如果這些是屬於固定顯示的部份，則要以大括號括起來，否則他會順序計算顯示。請試著和第 5.5.2 小節，頁 52 的標準 enumerate 環境比較一下。編譯後的結果如下：

http://edt1023.sayya.org/tex/latex123/example15.tex
http://edt1023.sayya.org/tex/latex123/example15.pdf

這裡請注意一下一些同名的環境、巨集套件，例如 array 巨集套件及 array 環境，這裡的

enumerate 巨集套件也是一樣。

7.3.16 fontsmpl

這是字型 sample 測試 package，他可以是互動的，也可以引用這個 package 後直接使用 \fontsample 這個指令來印出目前使用的字型 sample。

互動的話，要自行輸入字族名稱。sample 檔在 $TEXMF/tex/latex/tools 目錄下，只要下：

```
latex fontsmpl.tex
```

就可以了，他會出現：

```
This is TeX, Version 3.14159 (Web2C 7.4.5)
(./fontsmpl.tex
LaTeX2e <2001/06/01>
Babel <v3.7h> and hyphenation patterns for american, french,
german, ngerman, nohyphenation, loaded.
(/usr/share/texmf/tex/latex/base/article.cls
Document Class: article 2001/04/21 v1.4e Standard LaTeX document class
(/usr/share/texmf/tex/latex/base/size10.clo)) (./fontsmpl.sty)
Please enter a family name (for example `cmr').
\family=
```

只要輸入要測試的字型字族，例如 cmr，他就會產生 fontsmpl.dvi 這個檔，然後就可以使用 dvips 或 dvipdfm[x] 把他轉成 ps/pdf 格式的檔案。他只會測試 OT1 及 T1 兩種字型編碼。

7.3.17 ftnright

LaTeX 在兩欄式排版（two-column mode）時，他的腳註是置放在各自欄位底部。ftnright 會將兩欄式排版時，把所有的腳註都置放在右欄底部。這樣可以將腳註集中，看起來不會那麼凌亂。

7.3.18 indentfirst

通常，LaTeX 的章節開頭的第一個段落是不縮排的，在第二個段落起才會縮排。如果習慣每個段落都有縮排，可以使用 indentfirst package。這個套件也是引入就可以了，無需任何

指令。

7.3.19 layout

這是顯示目前版面配置的 package。引入這個 package 後，只要在本文區下 `\layout` 指令，他就會畫出目前的版面配置，也會將各種數據顯示出來。我們在第 5.2.1 小節，頁 43，裡頭所顯示的版面圖，就是這樣畫出來的。

7.3.20 multicol

在 LaTeX 宣告文稿類別的同時，我們可以選用 `twocolumn` 來選擇兩欄式的排版，再多則不行。在兩欄式的排版時，我們可以使用 `\onecolumn` 及 `\twocolumn` 指令，在單欄及兩欄間變換，但這有一個很嚴重的缺點，那就是欄位變換也會迫使換新頁，原來的頁面將會顯得空曠。

multicol 的目的，不僅突破兩欄，可以做多欄式的排版（最多可至十欄的排版），也可以在變換欄位編排時在同一頁面變換，而不必換新頁。他提供了 `multicols` 環境來做欄位的變換。他的使用方法很簡單，欄位數目及變換完全由環境來控制：

```
  ...
\usepackage{multicol}
  ...
\begin{multicols}{欄數}
  ...
  內容，依正常單欄方式書寫即可
  ...
\end{multicols}
```

請注意，引入時 multicol 是沒有 's' 的，而環境中的 `multicols` 是有 's' 的。multicol package 處理腳註的方式，和單欄排版相同，就是通通置於本頁底部，不分左右欄位。

7.3.21 rawfonts

這是模擬 LaTeX 2.09 舊版的低階字型指令，例如 `\texrm` 代表 10pt 的羅馬字族的字。在新版的 LaTeX 2_ε 並沒有定義這些指令。

7.3.22 somedefs

這是寫 LaTeX 巨集的一些範例定義，可以很容易的更改其中設定來寫自己的 package。這不在這篇文章的討論範圍，因此就不多談了。

7.3.23 showkeys

這個 package 會把 `\label`, `\ref`, `\pageref` 等交互參照的指令內容，或文獻引用內容，在指令所在處印出來。

7.3.24 varioref

這是加強型的交互參照的方式，我們會在第 11 章來討論。

7.3.25 verbatim

這是加強 LaTeX 原來的 `verbatim` 環境的同名套件。可以在裡頭使用註解，也可以利用 `\verbatiminput{檔案名}` 指令來引入外來檔案，當然，引入後會自動進入 `verbatim` 環境中。

7.3.26 xr

xr 是 eXternal References 的縮寫，意思就是交互參照外部的檔案。這會在第 11.2 節，頁 139 來討論。

7.3.27 xspace

這個 package 會在一個巨集結束時聰明的插入適當的空白。我們在第 3.3.2，頁 16，時曾談到 LaTeX 指令的結束的問題，所以，我們得在指令後加 '\ ' 或者 '{}' 來結束一個指令。但如果巨集定義時使用了 xspace 巨集套件的一個指令 `\xspace`，那麼他就會自動判斷指令何時結束，而不必自行插入 '\ ' 或 '{}' 了。

7.3.28 theorem

這是 LaTeX 內建的 theorem 環境的加強型巨集套件。我們會在第 10，頁 124，再來討論。

7.4 巨集套件何處尋？

重複發明輪子要盡量避免，所以，如果需要些功能，而 LaTeX 似乎沒有，那可以先找看看是不是別人已經有寫好類似的功能的巨集套件，首先要找的應該是 FAQ 文件：

http://www.tex.ac.uk/faq

在 news://tw.bbs.comp.tex 也有一篇 Chun-Chieh Huang 所維護的中文 FAQ 可以參考。另外 news://comp.text.tex 則是英文討論群組，可以多多利用。如果已經知道套件名或關鍵字，那可以到：

http://tug.ctan.org/CTANfind.html

去搜尋，搜尋到後可以抓整個目錄的壓縮檔。通常，安裝 package，他裡頭會說明如何安裝，萬一沒有的話，可以到 bbs/news 上發問，或者下載以下這個 sh script：

http://edt1023.sayya.org/tex/latex123/ltxins.sh

他的使用方法很簡單，把他置於執行路徑可及之處：

```
ltxins.sh your.dtx(or your.ins)
```

這樣就行了，他會產生必要的 *.sty 或 *.tex 及 pdf 格式的說明文件。這裡頭使用的是 dvipdfm[x]，所以要有安裝這兩種工具才行。當然，他不會自動安裝，你還是要手動把一些檔案拷貝到 LaTeX 找得到的地方。這只是一個很簡單的工具，因此不保證能成功編譯出文件出來。

如果想知道某個套件在系統上是否已安裝，安裝在什麼地方，可使用以下的小工具：

http://edt1023.sayya.org/tex/latex123/ltxpkg.sh

把他拷貝至路徑所及之處，使用方法如下：

```
ltxpkg.sh package-name[.sty]
```

這會列出所查詢的套件是否已安裝，及安裝在什麼目錄下，及這個 package 是否有預先載入其他的 package(s)。當然，這兩個小工具都是 bash script 寫的，你的系統要有安裝

bash，一般的 Unix-like 系統應該沒有問題，Mac OS X 及 Windows/cygwin 環境就不敢保證了。這也是個很簡單的小工具，如果是由 \input 指令所載入的其他.tex 檔，可能就會偵測不出來了，許多特殊細節也是沒有去考慮到，因此實際上要以 package 的原始 macro 內容爲準。以下爲一個使用實例：

```
edt1023:~$ ltxpkg.sh colortbl
The position of this package is at:
/usr/share/texmf/tex/latex/carlisle/colortbl.sty
The preloaded package(s) of `colortbl.sty' is(are):
array,color
```

所以，得知 colortbl package，系統上已經安裝，而且他會在引入的同時也引入 array 及 color 這兩個 packages，除非要變更這兩個 package 引入時的選項參數，不然就可以不必引入這兩個 packages 了。

第 8 章

表格的處理

這是屬於一般人覺得比較困難，但卻是很重要的部份，讓我們多花點時間研究。LaTeX 的表格，因爲是抽象邏輯的思考方式來製作表格，對一般使用者而言，比較不容易轉換成直觀印象。當然，有些編輯器，例如 GNU Emacs，有方便畫 LaTeX 表格的編輯器 script，但這些我們先不去理他，先從 LaTeX 本身表格的結構理解起，這樣在使用其他的輔助工具時也會比較得心應手，甚至沒有其他工具，只要把握住表格的大結構，製作表格就不會摸不著頭緒了。

由於 LaTeX 內建的表格功能算有點陽春，因此這一章會介紹一些外來的巨集套件，來彌補 LaTeX 表格功能的不足，這些巨集，使用上算相當普遍，幾乎所有的 TeX 的各種發行版本都會附上，因此不必擔心可攜性的問題。

8.1 表格的種類

表格的使用，在文章上常常是必備的要件，他有歸納及醒目的作用，當然，表格太多也是會喧賓奪主。通常，我們中文的使用習慣，表格就是大方框內有小方框，文字置於小方框內，甚至某些小方框內還有斜線在分隔。爲了排版上的方便及視覺表現上的美觀、清楚，在國際上大部份較正式的論文已不使用縱線、斜線，表格通常由橫線來做區隔，甚至完全沒有線條，使用空間區隔的方式。這種趨勢幾乎在二十幾年前就已開始普遍，只是國內的文件似乎還是很喜歡有縱、斜線在表格之中，好像沒有一些框線層層包住就不像表格。如非特殊的表現上的需求，我們應該朝簡化表格本身的方向走，將重點置於表格的內容及表格的邏輯結構安排，漂漂亮亮的表格外觀加上不當的內容配置，個人覺得這是個失敗的表格製作。

另外，等粗的雙線條，可能也是得盡量避免，通常粗細不等的外框雙線條有裝飾的作用，因此，如果文件是較正式的論文，那就可能要避免，如果是海報、DM 或要讓人們填寫的

表格之類的，那又是另外一回事，這時封閉性的方框可能會有需要。這些規範只不過是一些慣例，並非一成不變的，得視文件的性質及使用場合來做變化，一個大原則是，如果是以文字敘述爲主的文件，那麼，表格本身如果比文字內容搶眼太多的話，或許就要考慮簡化表格本身了。

我們這裡就來比較，有縱線、無縱線、完全沒有線框及含雙線表格的各種形式的表格，大家就自由心證，看哪一種表格看起來比較順眼。由於 HTML 格式在表格的表現上可能會失眞，因此這裡製作成 PDF 格式供參考：

http://edt1023.sayya.org/tex/latex123/test-tables.tex
http://edt1023.sayya.org/tex/latex123/test-tables.pdf

8.2 tabbing 環境

這是 LaTeX 裡頭最基本的表格形式，除非自行另外定義、繪製，他並沒有方便可用的線條指令來區隔，完全使用空間、位置的配置來顯示表格內容，這時整個 tabbing 表格在 LaTeX 的地位並不是一個最小單位的 box，LaTeX 不會把整個表格當成一個單位來處理。所以，tabbing 表格是可以跨頁的，他可以被分成兩半來處理。因此，要和其他文字、圖表並排排版時，得另外放進一個 box 中，讓他自成一個 box 單位，例如 \parbox 或 minipage 環境裡頭。

在 tabbing 環境中，第一個列（row）是以 \= 來標示 Tab 寬度來區隔欄位（column），這個寬度是由欄位裡頭的字串寬度所決定的。後續的每個欄位是由 \> 這個符號來區隔，每列尾要自行加上 \\ 來換行，最後一行可以不必使用 \\ 換行。tabbing 的基本大結構是：

```
\begin{tabbing}
column1 \= column2 \= column3 \\
item1   \> item2   \> item3   \\
itemA   \> itemB   \> itemC
\end{tabbing}
```

這裡特意把他排列整齊（事實上，不排整齊 LaTeX 也會幫忙排好），這樣才能看得出來他的表格結構。那如果想調整欄位寬度時可以使用 template 的方式，例如：

```
\begin{tabbing}
xxxxxxxxxx\=xxxxxxxxxx\=xxxxxxxxxx \kill
column1 \> column2 \> column3 \\
item1   \> item2   \> item3   \\
itemA   \> itemB   \> itemC
\end{tabbing}
```

這裡以 10 個 x 爲欄位的寬度，這裡的 \kill 表示這一行是不印出來的，只是在表示各個欄位的樣本寬度，而且他會自動換行。當然，要使用其他的字串也是可以，例如以表格中最長字串來取代整個 x 字串，這樣就會讓欄位寬度剛好都可以容納其他欄內內容。也可以使用 \hspace{6em} 或其他的長度指令，來指定欄位的寬度。

對於欄位內文字的控制，tabbing 較不完備，雖然 LaTeX 有提供 \' 讓這個符號之前的文字靠左，及 \` 讓這個符號之後的文字靠右，但實際運用，可能不是使用者想要的結果，因此 LaTeX 的表格，主要還是以 tabular 環境較爲常用。但 tabbing 環境的好處是，他不見得一定要用於表格的排版，例如他也可以表現如條列環境般的另一種表現方式，而且他可以跨頁排版。

8.3 tabular 環境

這大概是最常使用的表格形式，可以很方便的畫線框。這種表格，LaTeX 是把整個表格當成一個單位來處理，就像字母一樣，因此他在版面的安排上是和一般的字母一般的處理，所以，這種表格不經特殊處理，無法被分割成兩個部份來跨頁。

和 tabbing 環境的不同，除了可以有線條之外（tabular 環境，當然也是可以完全沒有線條），分隔欄位的符號是 &，而且，一定要指定欄內文字的置放位置，欄內文字超出指定的寬度時，會自動折行，還有許多其他更細節的調整。

8.3.1 tabular 表格的基本結構

```
\begin{tabular}[t]{lll}
\hline
column1 & column2 & column3 \\
\hline
item1   & item2   & item3 \\
itemA   & itemB   & itemC \\
\hline
\end{tabular}
```

其中 [t] 表示 top，也可以是 b 表示 bottom，或 c 代表 center，這要在前後有文字相並排的時候才會顯現作用，因爲 LaTeX 會把整個 tabular 表格當成一個字母單位，所以可以和其他文字、圖表並排排版。這些參數的意思是和同行文字的對齊方式，top 是表格頂端和前後文字對齊，bottom 則是表格底部和前後文字對齊，center 則是和表格中央對齊。

換行的方式和 tabbing 環境一樣，其中的 \hline 是畫一條橫線的意思，連續兩個 \hline\hline 會畫雙橫線，他本身會自動換行，因此不必加上換行符號。其中 \begin{tabular}{lll} 的 lll 是在指定各欄位內容在小方框內的置放位置，l 表示靠左（left），r 表示靠右（right），c 表示置中（center）。在 {lll} 中加上 bar（|）會畫縱線，例如 {|l|l|l|} 這樣就會變成傳統的大方框、小方框的表格。而兩個 bar 就會畫雙縱線。

tabular 環境內尚可使用另一個 tabular 環境來製作更複雜的表格，這在 tabbing 環境是不被允許的。

8.3.2 tabular 環境對欄位的調整

1. p{寬度}
   這裡的 p 指的是段落（paragraph）。通常用於一個小段落的文字，指定了寬度後裡頭的文字會自動折行，而且這個段落的頂端會和其他欄位的頂端對齊。

2. @{文字、符號或指令}
   這可以作用在本欄的各個列，讓他們都出現某個文字、符號或都在某個指令的作用下。這個指令另外會同時將欄位間距縮成 0，置於首尾的話，會有讓橫線和文字切齊的作用（預設不會切齊，橫線兩端會多出欄位間距的部份）。

3. \multicolumn{欄位數}{左右位置}{文字內容}
   跨欄排版，例如一小段文字跨兩欄。左右位置可使用 lrc 之一。

4. \cline{a-b}
   畫某部份欄位的橫線，其中的 a-b 指的就是要畫線的欄位數，例如 \cline{2-3} 就是畫第二欄至第三欄的橫線。

5. \arrayrulewidth=單位長度
   調整表格線條的粗細，預設值是 0.4pt。使用方法：\arrayrulewidth=1.5pt 即可，但要注意的是要在進入 tabular 環境之前設定好。

6. \tabcolsep=單位長度
   調整兩欄位的左右間距。請注意，這個值是實際兩欄位間距值的一半，預設是 6pt。使用方法和 \arrayrulewidth 一樣。

7. \doublerulesep=單位長度
   調整畫雙線時，這兩線間的間距，預設值是 2pt。使用方法和 \arrayrulewidth 一

樣。

8. \arraystretch
   調整表格的上下行距。請注意，這要由 \renewcommand 來重設，因爲在 LaTeX 定義出的一個常數值，而這個 \arraystretch 只是這些常數值的倍數，我們要重新改變他才能改變預設倍數。例如：example16.tex 中的使用方法。

在 tabular 環境的參數中，可能是取代原來的參數，例如 p{}。也可能是置放在原參數的前後，如 @{}，這看一下實際例子就可以瞭解：

```
% example16.tex
\documentclass{article}
\usepackage{textcomp}              % for \textcelsius
\renewcommand{\arraystretch}{1.2} % 將表格行間距加大爲原來的 1.2 倍
\arrayrulewidth=1pt                % 調整線條粗細爲 1pt
\tabcolsep=12pt                    % 調整欄間距爲 24pt
\begin{document}
\centering
\section*{SPECIFIC HEATS (20 \textcelsius\ AND 1 ATM)}
\begin{tabular}{@{\sffamily }lll@{}}    % 第一欄位使用 sans serif 字族
\hline
 & \multicolumn{2}{c}{\bf Specific Heats} \\ % 跨二三欄排版，文字置中
\cline{2-3}                                    % 只畫二三欄橫線
 & $c$ (J/kg$\cdot$K) & $C$ (J/mol$\cdot$K) \\
\hline
Aluminum     & 900  & 24.3 \\
Copper       & 385  & 24.4 \\
Gold         & 130  & 25.6 \\
Steel/Iron   & 450  & 25.0 \\
Lead         & 130  & 26.8 \\
Mercury      & 140  & 28.0 \\
Water        & 4190 & 75.4 \\
Ice ($-$10 \textcelsius) & 2100 & 38 \\
\hline
\end{tabular}
\end{document}
```

textcomp 也是 LaTeX 的標準巨集之一，他提供了許多符號，不必進入數學模式也是可以正常使用。但一般編譯的話，可能會使用到 pk 點陣字，如果有安裝 cm-super Type1 字型的話，可以使用以下的編譯方式：

```
latex example16.tex
dvisp -Pcm-super example16.dvi
ps2pdf example16.ps
```

這樣就會完全使用 Type1 字型。如果沒有安裝 cm-super Type1 字型，則可引用 txfonts 或 pxfonts 巨集套件。

@{} 如果完全沒有加入任何參數，那麼他的作用只是在去掉左右兩欄間距而已，大家可以把有關 @{} 的部份拿掉，試著再編譯看看，仔細比較看有什麼不同。有些專業排版的專家建議把表格前後加個 @{} 去除突出來的橫線（實際上就是去除原有左右兩邊間距的部份）。編譯好的例子在：

http://edt1023.sayya.org/tex/latex123/example16.tex
http://edt1023.sayya.org/tex/latex123/example16.pdf

如果 @{} 裡頭不是指令，而是文字或符號，那這個文字或符號會加在各欄文字內容的前或後。

p{} 指令的使用時機是某一個欄位的文字比較多，需限定欄位的寬度讓他自動折行的情形，例如以下的例子：

```
% example17.tex
\documentclass{article}
\renewcommand{\arraystretch}{1.2} % 將表格行間距加大爲原來的 1.2 倍
\begin{document}
\centering
\section*{Yi Syllables Area Character Blocks}
\begin{tabular}{@{}llp{6cm}@{}}
\hline
Start & End & Character Block Name \\
\hline
A000  & A48F  & Yi Syllables.
                Yi also known as Lolo, is a script resembling Chinese
                in overall shaps that is used in the Yunnan province
                China. \\
A490  & A4CF  & Yi Radicals.
                Basic units of the Yi syllables. \\
\hline
\end{tabular}
\end{document}
```

這樣會把 p{} 指定的欄位當成一整個段落來處理，空一個空白行，同樣是表示新段落的開始。編譯好的例子如下：

http://edt1023.sayya.org/tex/latex123/example17.tex
http://edt1023.sayya.org/tex/latex123/example17.pdf

8.4 array 巨集套件

這個巨集套件可以加強原有 tabular 環境的功能。使用上只要引入 array 巨集套件即可，tabular 環境依原來的使用方法，只是多了些相關調整指令。

1. m{寬度}
   這和 p{} 一樣的作用，只是置放的位置不一樣，此時其他欄位的內容會對齊這個段落的中央位置。

2. b{寬度}
   同 p{}，但其他欄位的內容會對齊整個段落的底部。

3. >{指令}
   這可以置於 l,r,c,p,m,b 參數之前，是對於某個欄位的內容下指令，這個指令會在此一欄位內容之前作用。引用了 array package 後，可能會抑制某些 @{指令} 的作用，此時要改用 >{指令}，但這沒有去除欄位間距的功能，可在前頭再加個 @{} 即可。

4. <{指令}
   和 >{指令} 相同，但會在此一欄位內容之後才作用。

5. !{指令}
   這是取代 | 的作用，可以方便使用特殊符號來代替原來的縱線。

6. \extrarowheight
   這是在調整欄位內容頂端的空間大小，但不會改變底部的空間大小。

8.5 tabularx 巨集套件

tabularx 巨集套件提供一個 tabularx 環境，這是加強型的 tabular 環境，附在 LaTeX 的工具組裡頭。主要作用是改善 \tabular* 指令，指定表格寬度的功能。

在原始 tabular 環境，加了個星號，可以指定表格的寬度。但由於 \tabular* 這個原始環境，他會去修改欄內空間，而不是實際整個表格方框的寬度，這使得某些文字會超出表格範圍，因此，使用上可能 tabularx 會比較方便，他提供了 X 參數來取代原來的 lrc 參數，這個參數實際的作用是 p{} 的功能，因此會實際調整欄位方框的寬度，而且裡頭的

文字敘述超過欄位寬度時會自動折行。這個套件會自動引入 array package[1]。這裡使用這兩種環境來排版，大家比較一下他的結果，就知道差異了：

```
% example18.tex
\documentclass{article}
\usepackage{tabularx}
\parindent=0pt
\renewcommand{\arraystretch}{1.2}
\begin{document}
\centering
\section*{\texttt{tabular*} environment}
\begin{tabular*}{8cm}{lll}
\hline
Start & End  & Character Block Name \\
\hline
3400  & 4DB5 & CJK Unified Ideographs Extension A \\
4E00  & 9FFF & CJK Unified Ideographs \\
\hline
\end{tabular*}
\section*{\textsf{tabularx} package}
\begin{tabularx}{8cm}{llX}  % 8cm 減去前兩個欄位寬度後，剩下的通通給
\hline                      % 第三欄位使用，文字超出的部份會自動折行
Start & End  & Character Block Name \\
\hline
3400  & 4DB5 & CJK Unified Ideographs Extension A \\
4E00  & 9FFF & CJK Unified Ideographs \\
\hline
\end{tabularx}
\end{document}
```

tabularx package 並不是都沒有缺點的，例如，使用 \verb 指令時會有一些不相容，另外，在 tabularx 環境內還要有其他的 tabularx 環境時，這個在裡頭的 tabularx 環境要由大括號括住，不能像 tabular 環境一下的直接巢狀使用。編譯好的例子在：

http://edt1023.sayya.org/tex/latex123/example18.tex
http://edt1023.sayya.org/tex/latex123/example18.pdf

8.6 表格線條粗細的控制（booktabs）

由前面幾節所述，可以看得出來 LaTeX 表格巨集的功能稍嫌陽春了點，對於一些特殊狀況可能會無法處理，對於表格外觀要求較高的使用者也會感到不足，雖然也可以自行去定義巨集，但這樣一來不但可能有可攜性的問題，而且也不是每個人都有時間去學習

[1]套件查詢，可使用第 7.4 節，頁 74，所提到的 ltxpkg.sh 來查詢是否有預先載入其他的 packages。

TEX/LATEX 巨集的寫作。我們試圖來看看有沒有其他的解決方式，這裡不得不會提到一些外來的巨集套件，但這些套件的使用相當的普遍，幾乎可以忽略他的可攜性的問題。

我們前面曾學過 \arraryrulewidth 指令，可以調整線條的粗細，但是這無法各別調整線條，每個在 tabular 表格環境內的線條會調整成一樣的粗細。booktabs 巨集套件可以很方便的達成這個目的。我們來看看這個提供了什麼方便的指令：

指令	功能
\toprule[線條粗細]	畫表格頂端的橫線
\midrule[線條粗細]	畫表格裡頭的橫線
\bottomrule[線條粗細]	畫表格底部的橫線
\cmidrule	指令某個欄位畫橫線，取代原來的 \cline

使用方法和 tabular 環境差不多，連環境名稱都一樣，但可在指令後加個方括號來指定線條的粗細，不指定的話，toprule 及 bottomrule 都會比中間的其他線條粗一點。其中 cmidrule 另有更進一步的功能：

```
\cmidrule[線條粗細](左右是否去邊){畫線欄位}
```

其中「畫線欄位」和 \cline 一樣，指定欄位數即可，例如 2–3。左右去邊要表明左（l）或/及右（r），也可由大括號指定要去掉多少（預設 0.5em），如：(lr{0.7em}){2-3}。我們把 example16 拿來改一下，大家試著看看有什麼不同，編譯好的例子如下：

http://edt1023.sayya.org/tex/latex123/example19.tex
http://edt1023.sayya.org/tex/latex123/example19.pdf

由於螢幕解析度的關係，如果分不出不同，請由印表機印出來比較，或將檔案放大再來觀察。這裡最粗的是 toprule 及 bottomrule 再來是 midrule，最細的是 cmidrule。而且 booktabs 已經調整過原來 tabular 表格的上下間距，除非想得更大，不然的話，不需另外再去設定 arraystretch 的值了。

8.7 彩色表格（colortbl）

彩色表格已經是很普遍，但千萬要小心喧賓奪主的情況，也別弄成了大花臉。因此，淡色系可能會比較合適。我們在第 3.4.3.1 小節及 example13 曾提到過 color package 的引用，但並沒有詳細說明這個套件的用法，而 colortbl 會使用到這些顏色的功能，因此這裡稍微說明一下。

8.7.1 color 巨集套件

這是附在 LaTeX 工具組 graphics package 中的一個巨集，使用上非常簡單，只要把 color 巨集在文稿 preamble 區引上就可以使用顏色了。以下是常要用到的控制指令：

指令	作用
`\color{顏色}`	這會使用文章所有內容都使用這個顏色
`\definecolor`	定義顏色
`\textcolor{顏色}{文字內容}`	讓文字內容使用某特定顏色
`\pagecolor{顏色}`	這是在設定背景顏色，本頁及其後的頁面會使用這個背景顏色
`\normalcolor{顏色}`	回復原來的顏色
`\colorbox{顏色}{文字內容}`	這是方框背景的顏色
`\fcolorbox{框色}{框內背景色}{文字內容}`	這是方框顏色和其內背景顏色不同

這裡要注意的是，指令裡頭使用的顏色，必需是有定義過的顏色才能使用。color 巨集只定義了一些基本顏色，red, green, blue (RGB 模型原色), cyan, magenta, yellow, black (CMYK 模型原色), white，另外一個常用的 gray 灰階模型（gray-scale），其他的顏色得自行定義。定義顏色的語法如下：

```
\definecolor{顏色名稱}{色彩模型}{調色盤值}
```

第一個參數就是自訂的一個顏色名稱，色彩模型可使用 rgb、cmyk 或 gray，各顏色深淺值在 0–1 之間，「調色盤值」就是各種原色的值。RGB 顏色的索引值，如果是 Unix-like 系統，可找一下 `rgb.txt` 這個檔案，裡頭就會有各種顏色的索引值，或者，參考 example24.pdf。這裡以 bisque 這個顏色爲例子，他的 rgb 三原色的深淺比例爲 255, 228, 196，各除以 256 得 0.996, 0.891, 0.755，定義方法如下：[2]

```
\definecolor{bisque}{rgb}{.996,.891,.755}
\definecolor{mypink}{cmyk}{.1,.8,.4,.1}  % cmyk 模型的例子
```

這樣以後就可以使用 bisque 及 mypink 這兩種顏色了。gray 則支援灰階，可以增減他的顯現深淺，例如：

```
\definecolor{mygray}{gray}{.6}
或直接定義及使用，不事先定義好顏色名稱：
\textcolor[gray]{.3}{文字內容}
\textcolor[rgb]{.2,.5,.7}{文字內容}
```

[2]每一個原色在電腦上最小可由一個 byte(8-bits) 來儲存，共有 256(2^8) 種變化，所以，三原色可調出 256^3 共 16,777,216 種顏色，但這不表示你的螢幕、印表機有辦法顯示那麼多顏色。

這樣 mygray 會得到淺灰色的效果，他的顏色名稱就是 mygray。直接定義及使用雖然也可以，但不建議這麼做，因爲如果有兩個地方要使用同一種顏色時，又得重複定義一次。要注意的是 gray 不能直接使用，要先定義他的灰色度，其他顏色也不能這樣單純靠一個值來定義他的深淺度。通常我們引用的時候，會加入以下的選項參數：

```
\usepackage[usenames,dvipsnames]{color}
\usepackage{colortbl}
```

這樣就可以使用 dvipsnam.def 這個檔裡頭所定義好的顏色，例如 Salmon, Orchid, BlueViolet…… 等等，請自行查閱這個檔案內容，dvipsnames 這個參數也可以不用，只用 usenames 即可。這裡引用時請注意順序，colortbl 要在後面，原因是 colortbl 巨集會自動引入 color 及 array 這兩個巨集，但裡頭並沒有含任何選項參數，所以要搶先去宣告。

8.7.2 colortbl 的主要指令

指令	作用
\columncolor	讓整個欄位著色
\rowcolor	整整個橫列著色
\arrayrulecolor{顏色}	指定線條的顏色
\doublerulesepcolor{顏色}	指定雙並線內間隔的顏色

在這裡，\columncolor 和 \rowcolor 的參數是一樣的，他們的共同語法是：

```
\columncolor[色彩模型]{顏色}[左緣突出長度][右緣突出長度]
```

我們現在就來看個實例，這裡頭有四個小例子，包括：灰階橫條、部份欄位著色、整個表格在著色背景及單一個表格內方框著色：

http://edt1023.sayya.org/tex/latex123/example20.tex
http://edt1023.sayya.org/tex/latex123/example20.pdf

8.8 表格的註解（threeparttable）

表格的註解比較麻煩，LaTeX 把 tabular 環境視爲一個單位，對裡頭的文字做腳註的話，將會不翼而飛，有些巨集套件有辦法在表格內做腳註（例如 longtable package），但卻是置於頁面底部，和一般內文的註腳混在一起，多數使用者希望的是能把註解就置於表格底部。解決的方法就是使用 threeparttable package，暫時將表格的某部份分割出來。

如果你的 threeparttable package 的表現和這裡的例子有不一樣的情形，請更新這個套件，這篇文章使用的是 2003/06/13 v3.0 的版本。

ftp://ctan.unsw.edu.au/tex-archive/macros/latex/contrib/misc.zip

下載後解開壓縮檔，把 threeparttable.sty 拷貝至 $TEXMF/latex/misc 目錄下，執行一下 texhash 一下即可。他的環境名稱和套件名稱一樣，就是 threeparttable。把 tabular 及註解的指令和內容，通通包在 threeparttable 環境裡頭即可。

他是使用 \tnote{符號或文字} 先標出要註解的地方，在 tabular 環境結束後，再使用 tablenotes 環境來寫註解內容，這兩個部份都是整個被 threeparttable 環境包住的。底下這個例子來自這個套件的作者 Donald Arseneau，這裡把他和 booktabs package 結合起來用：

http://edt1023.sayya.org/tex/latex123/example21.tex
http://edt1023.sayya.org/tex/latex123/example21.pdf

要注意的是，在 tablenotes 環境下，字體並沒有縮小，可參考 example21 裡頭，使用 footnotesize 的字體大小。

8.9 小數點對齊（dcolumn）

這是 LaTeX 的標準巨集，用於將表格內的小數點對齊。原來的 tabular 環境的作法是去增加一個欄位，那個欄位使用 @{.} 來專門排小數點，這樣一來兩欄的間距會消掉，看起來就像連在一起的數字了，但這樣實在是有點 dirty，使用 dcolumn 巨集的話，就可以很有規律的去對齊小數點或逗點。

dcolumn 的用法，主要是去取代 tabular 參數中的 lrc 這些參數。他使用的是一個大 D 指令，後接三個參數：

```
D{文稿輸入符號}{排版後輸出之符號}{小數位數}
```

例如以下的表格我們再怎麼去排，小數點總是無法對齊，因爲 tabular 環境是以整個字串在處理的：

```
\begin{tabular}{lllll}
\toprule
      & headA & headB & headC & headD \\
\midrule
test1 & 7.879  & 921.661 & 1382.81 & 998.98 \\
```

```
test2 & 1.97   & 35.21   & 321.3   & 4791112.11 \\
test3 & 211.97 & 5.2     & 213.629 & 748261594.106 \\
\bottomrule
\end{tabular}
```

我們只要把 tabular 的後面參數改成：

```
  ...
\usepackage{dcolumn}
  ...
\begin{tabular}{lD{.}{.}{3}D{.}{.}{3}D{.}{.}{3}D{.}{.}{3}}
  ...
```

就可以讓小數點對齊。這個 3 就是最長的小數位數，我們輸入、輸出都是英文句點（就是小數點），這樣的表示法也可以另外宣告 \newcolumntype 的標準格式，以簡化 tabular 參數的輸入，即：

```
  ...
\usepackage{dcolumn}
  ...
\newcolumntype{z}[1]{D{.}{.}{#1}}  % 定義一個新的 z 指令
  ...
\begin{tabular}{lz{3}z{3}z{3}z{3}}
  ...
```

那個 #1 就是 z 這個新指令的參數（z 可以是任意的字母或符號），z{3} 其實就是代表 D{.}{.}{3}。中括號裡頭的 1 代表這個新的 z 後面只接一個參數，在這個例子裡就是小數點個數 3。以下是編譯好的例子：

http://edt1023.sayya.org/tex/latex123/example22.tex
http://edt1023.sayya.org/tex/latex123/example22.pdf

這裡要特別注意的是，在 dcolumn 的效力範圍裡頭，例如以上例子，受 z 指令影響的欄位，他會自動進入數學模式，裡頭要表現數學式的話，前後不必再加 $，否則會跳出數學模式。例如 example22 裡頭，那些 headA 會變成斜體字，這是因爲進入了數學模式，要讓他正常的話，就要寫成 $headA$ 這樣來跳出數學模式。

8.10 大型表格（longtable）

這可能有兩種情形。一種是很寬的表格，另一種是很長的表格。太寬的表格可考慮旋轉一下，讓他橫放，至於長的表格可以使用 longtable 讓他可以跨頁連續。如果都不行，那只考慮夾頁，圖表另外製作，或者試著簡化圖表一途了。

8.10.1 太寬的表格

要把表格橫放，方法很多，例如 graphics package 一起 release 的 lscape 巨集套件，他會讓內文旋轉九十度，或者使用 graphics/graphicx package 本身的 `\rotatebox` 指令，將表格旋轉九十度。另外，也可以使用 rotating package 來旋轉。這些 package 基本上使用的都是 graphics/graphicx 巨集上的旋轉指令的功能，所以，不限定只能使用在圖表而已。

這裡就以 rotating package 爲例來說明，他提供了 `sidewaystable` 及 `sidewaysfigure` 環境，前者會讓表格旋轉九十度，後者會讓圖形旋轉九十度。這個套件會自動引用 graphicx 巨集，不使用這些套件，使用 graphicx 的 `rotatebox{90}{表格}` 也是可以達到相同的功能，只不過限制會比較多。這裡舉一個 rotating 的例子，把表格置於 `sidewaystable` 環境內就行了：

http://edt1023.sayya.org/tex/latex123/example23.tex
http://edt1023.sayya.org/tex/latex123/example23.pdf

8.10.2 太長的表格

表格想跨頁，可以使用 `tabbing` 表格，如果想使用 `tabular` 表格，又想可以跨頁的話，可以使用 longtable package，這是 LaTeX 所附上的工具組。他提供了 `longtable` 環境來取代原來的 `tabluar` 環境。如果想要和 booktabs 合用的話，請更新 booktabs package 的版本，目前最新的版本是 2003/03/28 v1.618：

ftp://ctan.unsw.edu.au/tex-archive/macros/latex/contrib/booktabs.zip

我們把前面提到過的 `rgb.txt` 拿來排成表格參考，例子如下：

http://edt1023.sayya.org/tex/latex123/example24.tex
http://edt1023.sayya.org/tex/latex123/example24.pdf

8.11 浮動環境

`tabular` 表格，LaTeX 都會把他視爲一個獨立的 box，也就是會把他當成一個字母單位在處理，他不能被分割，常常因爲圖表稍大些 LaTeX 就會起新頁去置放，但這樣一來原本的頁面就會顯得空盪，整個版面看起來很不自然，這種情形下，他們的置放位置就很重要了，使用浮動環境的話，LaTeX 會繼續文字的部份，而把圖表置放在下一頁的頂端。通

常，在 LaTeX 的浮動環境下，圖表通常會置放在一頁的頂端或都是底部，正常是不置放在一頁中間的位置，除非強迫指定，有放不下的情形時，就會讓他佔一整頁。因此，LaTeX 就得把前後位置經過整體的計算後再來決定圖表應該置放在什麼地方，這就是所謂的浮動環境。[3]

8.11.1 基本的浮動環境

LaTeX 的浮動環境很簡單，就是把表格置於 table 環境當中就可以了。在裡頭有 \{caption} 指令可以指定表格的標頭，而且編譯後會自動標上 'Table n:' 字樣，後接 caption 的內容，那個 n 會自動編號。

一般國際上較正式的文件，caption 置放的位置慣例是「表上圖下」，也就是說表格的標題是置於表格上方，圖形則在下方。但 LaTeX 對 caption 的置放位置，是對於不管圖表皆置於下方的配置，我們把 caption 置於上方時，caption 和表格的間距將會太小。如果不想手動去調整，可以找 topcapt package 試試看。手動修改的方法如下：

```
  ...
\begin{table}  % 進入浮動環境
\begin{table}[置放位置選項]
\setlength{\abovecaptionskip}{0pt}
\setlength{\belowcaptionskip}{10pt}
\caption{表格的標題}
\begin{tabular}{表格參數}
  表格內容
\end{tabular}
\end{table}
```

原始的定義，abovecaptionskip 和 belowcaptionskip 的值剛好相反。如果是兩欄排版時，要加個星號，\begin{table*}...\end{table*}。

8.11.2 浮動環境選項參數

LaTeX 的浮動環境的配置，有時會不符和我們實際上的期望，這時可加入選項參數。

[3]這篇文章並不使用浮動環境，因爲這篇文章的原先構想是一種講義形式的文件，並非一份正式的文件格式。所以這篇文章中，圖表並無 caption，他們的位置是當做一般文字內容排版的，爲了敘述的連續性，圖表都可能置於一頁中央。

位置選項	作用
h(here)	置於下指令處位置
t(top)	置於一頁的頂端
b(bottom)	置於本頁底部，如空間不夠會置於次頁
p(page)	單獨佔一頁，此頁沒有內文的部份
\suppressfloats[位置]	抑制浮動物件置放於本頁的某處，他會出現在次頁
!	置於以上選頁之前，會更強烈要求達到此選項的作用。但對 p 則無作用

如果都沒有指定，那預設是 [tbp]。其中 \suppressfloats[位置] 只能指定 t 或 b，不能同時指定好幾個，而且! 的作用會優先，他會忽略 \suppressfloats 的指示。這些指示，LaTeX 經過整體計算後，如果覺得窒礙難行的話，仍然會「抗命行事」的，畢竟他要以大局爲重。

第 9 章

圖形的處理

TEX 系統發展的時代，對於圖形處理還比較落後，當時，ps/eps/pdf、jpeg/png 這些圖檔都還不存在，因此這是 TEX 本身的一個盲點，雖然有 METAFONT/METAPOST 這些強大的造字、繪圖的程式，但這些工具，一般人恐怕不容易駕馭，因此，可能需要尋求更方便的外來工具。

但 TEX 聰明的地方就是，他本身不能處理的，就預留個位置，讓其他輔助的工具來處理，所以，這也是 TEX 20 幾歲了，還是能和新的工具配合的原因。圖形倒是還好，因爲有方便的巨集及繪圖工具來處理，只要能畫的出圖來，一切都好說，而繪圖的技巧就和 TEX/LATEX 本身的排版技巧不算是直接相關了。

9.1 圖形的種類

我們使用的圖檔，基本上分成兩大類，一是向量圖，不會因縮放而失眞，一是點陣圖，會因爲縮放而失眞，但視使用場合，並不是所有的圖檔都適合製作成向量圖的。不管是哪一種圖形格式，都是數位化的結果，在電腦裡頭儲存的都是數字，只不過解釋過程不同而已。由於製作高品質的文件通常都使用向量圖，因此，我們將會把重點放在向量圖，尤其是 eps/pdf 格式。

9.1.1 點陣圖形

這種圖形應該是佔最多數的，使用也最廣泛。他是使用自然的方式來儲存數位資料，把圖形所佔的頁面想像成是許多很細小的方格子所組成，每一個小格子就代表了一個圖素（pixel），這個圖素可能代表者各種不同的顏色，只要單位小格子愈多（解析度愈高），我們人類眼睛就會分辨不出其中的各小格式間的區隔，於是影像就可以平滑的顯示出來了。

我們平常常見的圖檔格式，例如：jpeg, gif, bmp, ico, xpm, png, psd, tiff····· 等等都是屬於點陣圖檔。由於他是由固定大小的圖素實際數位化儲存的，所以如果他們放大或縮小，我們的眼睛就會分辨出不同，甚至放得更大些，還可以看得出「格子」出來（這會造成所謂的鋸齒狀，jaggies），原來的影像就因此失眞了。

9.1.2 向量圖形

向量圖檔儲存的並不是實際各種圖素的資訊，而只是儲存數學運算的基本描述，顯像時再馬上計算出結果來顯示。例如，以一個圓形圖來說，他的圖檔可能只有儲存圓心所在、圓的半徑、顏色索引值等資料，要顯示時，馬上計算，然後顯像（在螢幕或印表機上，最後顯像當然仍是要轉成點陣圖的），但由於每放大、縮小時都會重新計算過，所以，就不會造成失眞了。當然，這會更耗電腦資源，但以目前的電腦軟、硬體進步的情形，這些消耗都可以控制在可被忍受的範圍。

目前最常見到的向量圖檔，應該就是 eps/pdf, svg····· 等圖檔，向量字型也是屬於特殊格式的一種向量圖。一般比較規律性結構的圖，會比較適合使用向量圖，自然界的實體影像可能就比較適合使用點陣圖了，但科技不會把腳步停下來，將來的數位化會是怎麼樣的情形就只能用我們的想像力去填補了。

9.2 繪圖工具

繪圖的工具實在是太多了，這裡不可能一一介紹，只能擇要的簡單說明。我們的重點是排版，因此要知道的是圖形怎麼安置於版面裡頭才會使整個版面協調一致，而不是在繪圖教學。就請大家自行選個順手的繪圖軟體去熟悉，這類工具，大概都是一理通百理通，畫筆怎麼用簡單，要畫出像樣的圖出來會比較困難。

9.2.1 原生繪圖工具

這是安裝 TEX/LATEX 系統，不管是哪一種的發行版本都會附上的，但可能會有不易入門的感覺，一旦抓到到了訣竅，這是不假外求的工具。因此，在這篇文章裡頭，會對這些原生的繪圖工具多做一些說明。這裡所提到的原生繪圖工具，另外一個好處就是可以使用 CJK 環境，意思當然就是說可以在圖中插入中文及 LATEX 排版後的結果，這恐怕是許多使用者希望的功能，但一般 GUI 式的繪圖工具就常常無法完整支援了。

1. LaTeX 的 picture 環境

   在 LaTeX 裡頭有個標準內建的圖形環境，那就是 picture 環境，但他只能繪製一些簡單的圖形，後來也有人寫了 epic 及 eepic package 來增強 picture 環境的功能，這可說是 LaTeX 「原生」的繪圖巨集，雖然功能不是很強，而且是由指令來指揮繪圖，不容易直觀的轉換過來，但好處是他是和 LaTeX 文稿結合在一起，使用的是 LaTeX 的指令，不是另外引用外來的現成圖檔，和 LaTeX 的結合當然會比較好。所以，我們在此會加以簡單介紹。由於 eepic 引用了 PostScript 的指令，使用 pdflatex 時會無法編譯，因此這章只會探討 picture 環境及 epic package。我們會在第 9.3 節做進一步的說明。

2. PSTricks 巨集

   另一個相當有名氣的繪圖巨集組 PSTricks package，功能就相當強了，他仍然是使用了 PostScript 的指令，所以，在 pdflatex/dvipdfm[x] 常會無法編譯，要 dvips 才有辦法解讀。但另有人寫了 PDFTricks package，可以轉換成 PDF device 認得的指令，所以，在此也會一併簡單介紹他們的使用。我們將會在第 9.4 節做進一步的介紹。

3. METAPOST 繪圖工具

   在 TeX 系統，則有 METAFONT 及 METAPOST 可供繪圖，這可說是 TeX 系統的標準繪圖語言，但和 TeX 的語法有很大的不同，是一種 object-oriented 式的 macro 語言，功能相當的強大，甚至可以製作字型。METAPOST 是 METAFONT 的改良版本，主要是讓預設的輸出是 eps 向量圖檔[1]，而且可以連續處理多個檔案，也可以嵌入 TeX/LaTeX 的語法在裡頭。

   目前 Knuth 教授編寫 TAOCP 使用的繪圖工具就是 METAPOST，我們這裡就不探討 METAFONT，在語法上 METAPOST 和 METAFONT 是類同的。由於 METAPOST 所產生的 eps 圖檔，不管是 latex 或是 pdflatex 都可以順利引用，所以將會在第 9.5 節獨立做進一步的說明。

當然，以上的任一個工具，要詳細說明的話，都可以寫成一本獨立的書，所以，在這裡只能簡單的介紹，沒有提到的部份，可以參考他們的使用手冊。METAFONT 及 METAPOST 則可以另參考 Knuth 教授所寫的 *The METAFONTbook*。

[1]嚴格來說，METAPOST 所產生的圖檔只是 eps 圖檔的一個子集，稱爲 mps，我們這裡一併稱他爲 eps 圖檔，不做嚴格區分。

9.2.2 外來繪圖工具

我們也可以從其他的外來繪圖工具來繪製圖形，然後再引入圖檔即可，這樣一來就可以使用自己熟悉的繪圖工具。繪圖的話，當然是以向量圖爲優先考慮，因爲他不會因爲縮放而失眞。但像一些照片類的圖檔，就不太適合使用向量圖了。

以下列出的都是 free 的繪圖工具，一般用途上，功能上不見得會輸商業產品，但使用界面上就不一定比商業產品方便。以下只做簡單的介紹，至於操作方法，請直接參考其中的使用手冊的說明。種類很多，請別眼花撩亂了，就選個一、二種去熟悉他吧！畫筆可能很多種，但畫圖「心法」只有一種。

1. gnuplot

   這是個有點古老，但卻非常實用的 XY 及 XYZ 數據資料及數學函數的繪圖工具，他有內建的繪圖語言來繪圖，可以使用交談式的方式，或寫成檔案來做批次處理。如果有安裝 kile 的話，他有 GUI 的圖形界面可以用來方便繪圖。他可以輸出 fig 圖檔，供 xfig 做進一步的修改、編輯，也可以輸出 LaTeX 的 `picture` 環境文稿及 METAPOST 程式碼供引入。幾乎主流的作業系統都有他的版本。他的網站及和 LaTeX 結合的一些 sample，可以參考：

   http://www.gnuplot.info/
   http://www.fnal.gov/docs/products/gnuplot/tutorial/
   http://cips02.physik.uni-bonn.de/~baehren/tex/gnuplot.html

2. GNU plotutils

   這是和 gnuplot 類似的繪圖工具及函式庫（GNU libplot for C, libplotter for C++），主要用於繪製 2D 科學數據及數學函數向量圖。他也支援 xfig 的 fig 圖檔。而且有現成的函式庫，對於寫繪圖程式的人來說也很方便，像後面會談到的 pstoedit 就有利用到這個函式庫。他的網站在：

   http://www.gnu.org/software/plotutils/

3. xfig

   這也是很古老的 X Window System 下的繪圖工具，他的檔案格式是公開的，所以 gnuplot 也支援他。他相當於 MS Windows 下的 CorelDraw，預設的輸出格式是 eps 圖檔，但也可以輸出 LaTeX `picture/epic` 文稿。請參考他所附的文件來和 LaTeX/gnuplot 配合使用。他的網站在：

   http://www.xfig.org/

   如果你的平台無法編譯 xfig，可以試試 Java 的 jfig：

http://tech-www.informatik.uni-hamburg.de/applets/javafig/

4. tgif

   這是和 xfig 類似的向量繪圖工具，也可以輸出 eps/pdf 圖檔供 LaTeX 來引入，gnuplot 也支援 tgif 圖檔，這個工具和 gif 圖檔是沒有關係的。記得，在好幾年前接觸 tgif 時，有人把他拿來畫卡通影像，效果還不錯，當然，他的主要用途不是在畫卡通。他的網站在：

   http://bourbon.cs.umd.edu/tgif

5. grace

   這是源自於古老的 ACE/gr 的 Motif 版本 xmgr[2]，由於改變版本爲 GNU GPL 發行，所以名稱改爲 grace，意思是 “GRaphing, Advanced Computation and Exploration of data”，要說是 “Grace Revamps ACE/gr” 也是可以啦！這是類似 gnuplot 的 X Window System 下的繪圖工具，但有漂亮的 GUI 操作界面，是 WYSIWYG 的 2D 數據資料繪圖工具，他需要 Motif 或 LessTif 函式庫，目前尚有少數 xmgr 原來的功能還沒有完全移植過來。他的網站在：

   http://plasma-gate.weizmann.ac.il/Grace/

6. GNU Dia

   這很適合拿來畫流程圖、電路圖的一個 X Window System 下的繪圖工具，使用 Gtk+ 函式庫，類似 Windows 下的 Visio。他可以輸出 eps/svg 及一般常見的點陣圖檔，也可以輸出 METAPOST、LaTeX PSTricks 及 xfig 的圖檔。

   http://www.lysator.liu.se/~alla/dia/dia.html

7. Ipe

   這是一般性的 X Window System 及 Windows 下的繪圖軟體，使用 Qt 函式庫。他的特點是，主要輸出 pdf 圖檔，並可嵌入 TeX/LaTeX 文字，也就是說圖裡頭的文字可以是 TeX/LaTeX 排版出來的結果，也可以輸出 eps 圖檔及 XML 檔。另外一個好處是，只要是 Ipe 製作出來的 pdf/eps 圖檔，他可以由 Ipe 重新載入後予以再次的編修，這對於 eps/pdf 檔案的編修非常的方便，這類向量圖檔的編修，一般是比較困難的，尤其是 pdf 格式。他的網站在：

   http://ipe.compgeom.org/

[2]原始的 ACE/gr 有兩種版本，除了 Motif 的 xmgr 外，還有另一個 XView 的 xvgr 版本，但 XView Widget 和 Motif 之爭，最後是 Motif 勝出。

8. skeneil(sketch)
   這在以前，稱爲 sketch，是一個一般性用途的繪圖軟體，可以輸出多種向量圖，包括 svg。他是使用 Python script 語言所寫的。他的網站在：
   http://sketch.sourceforge.net/index.html

9. GIMP(GNU Image Manipulation Program)
   這是一般性的繪圖軟體，有 X Window System 及 Windows 的版本。但他主要處理的是點陣圖，類似 Photoshop 軟體。他的網站在：
   http://www.gimp.org/

10. MetaGraf
    這是 METAPOST 的圖形界面軟體，這樣就可以使用 GUI 的方式來使用 METAPOST 的強大繪圖功能了。他是 Java 語言所寫的，所以要先安裝 Java 相關工具組才行。而且，如果是像製作字型這類精確度要求很高的圖檔的話，可能就不是很合適了。他的網站在：
    http://w3.mecanica.upm.es/metapost/metagraf.php

11. OpenOffice.org
    這是整合式的文書處理軟體，裡頭也附有繪圖工具 OpenOffice.org Draw(oodraw，可單獨拿做繪圖工具)。他的網站在：
    http://www.openoffice.org/

12. KOffice
    這是另一個整合式的文書處理軟體，裡頭附有許多不同用途的繪圖工具，例如：Kivio、Karbon、Krita 及 KChart 等，他們都可以單獨拿來繪圖。他的網站在：
    http://www.koffice.org/

9.2.3 圖形轉換工具

由於 dvips 只能接受 eps 圖檔及 PostScript 指令，而 pdflatex 則只能接受 pdf/jpeg/png 圖檔，dvipdfm[x] 除了可接受 pdf/jpeg/png 圖檔外，大部份情形下，也可以接受 eps 圖檔，但硬生生插入的 PostScript 指令，則仍然是無法接受，畢竟他是要生成 pdf 檔的，除非是由 METAPOST 所製作出來的圖檔，不然的話，就得使用轉換工具將他們轉成可被接受的格式，所幸，這方面的工具還算方便。上一節所提到的繪圖工具，在某種程度上也是可以轉換圖檔格式。

1. ImageMagick

   我們主要是使用這個軟體中的一個工具：convert，他是個功能非常強大的圖檔轉換程式，但主是用在點陣圖，縱使轉成向量圖也不是眞正的向量圖，只是把點陣圖 wrap 進向量圖檔裡頭而已，放大時仍然會有鋸齒狀。他的網站在：

   http://www.imagemagick.org/

2. netpbm

   這是許多圖檔轉換的小工具所組成的圖檔轉換工具組，主要是用在點陣圖的轉換。他的網站在：

   http://netpbm.sourceforge.net/

3. pstoedit

   這是眞正各種向量圖格式之間的轉換工具。向量圖的轉換，主要是利於編修，例如 eps/pdf 圖檔要直接編修的話，一般工具會有困難，如果我們轉換成 fig 圖檔，然後再交給 xfig 去編修；或轉成 METAPOST 原始碼，使用編輯器進行編修，完成後再轉回 eps/pdf，這樣就很方便了。他的網站在：

   http://www.pstoedit.net/pstoedit/

4. ps2eps

   通常我們手上的圖檔不一定是 eps，而是一般的 ps，也就是說除了眞正圖的部份外，尚有一些空白在圖的四周，這代表裡頭的邊界（BoundingBox）沒有定好，這樣引入圖檔的時候，除非另做其他處理，不然的話，連原圖周圍的不必要空白也會引進文稿裡去，通常我們只是想要有圖的部份，這時可以將這個 ps 檔經由 ps2eps 處理過，去除不必要的空白。這個程式的作者是 Roland Bless，使用 perl 所寫一個很實用的小工具，在 Windows 系統的話，只要有安裝 perl 及 GhostScript 也可以使用，作者也提供了一個 ps2eps.bat 批次檔供使用。他的網站在：

   http://www.ipv6.tm.uka.de/~bless/ps2eps

   系統上安裝 Ghostscript 的話，他也會附上一個 ps2epsi，這個工具也可以利用，但有時會算錯 BoundingBox 就是了。

9.3 picture 環境

由於這是 LaTeX 內建的繪圖環境，最能配合 LaTeX 原來的語法及版面配置，因此我們多花一點時間研究，使用時請另外引入 epic package，這樣可使用多一些繪圖功能。要注意的

是，LaTeX 的 picture 環境，和座標息息相關，所以，繪圖之前一定腦海裡要有個座標圖來定位，而且要有相對長度的想像。

9.3.1 進入 picture 環境

進入 picture 環境的方式就像進入其他的環境一樣，但他要指定圖形物件的大小：

```
  ...
\usepackage{epic}
  ...
\begin{document}
  ...
\begin{picture}(寬, 高)(參考原點)  % 進入圖形模式
  這裡下繪圖指令，形成一個或多個圖形物件，也可以寫入一般文字
  讓 latex 去排版。
\end{picture}
  ...
\end{document}
```

指定長、寬等度量時，可以加上單位，如果不加單位，事先也沒有指定使用單位，那就是以 pt 爲單位，(寬, 高) 是不能省略的，這在座標圖上，就是建立了左下角 (0, 0) 至右上角 (寬, 高) 的參考座標系。(參考原點) 指的是左下角的原點平移至這個位置，往後就以這個點爲原點，這個可以省略，省略的話，原點位置就是 (0, 0)。通常我們都會在進入 picture 環境前先加以指定好單位，例如：

```
  ...
\unitlenght=1mm          % 指定 picture 環境內的度量單位爲 mm
\begin{picture}(50, 50) % 要進入 picture 環境前指定
  ...
```

這樣在 picture 環境裡頭就無需使用單位，直接寫數字就可以了，而單位就是 mm。

9.3.2 picture 環境的繪圖指令

在繪製任何線條之前，我們通常會指定開始的位置，否則通通會從（參考）原點開始畫起。原則上，picture 環境內，有方向性的圖形物件的參考原點，例如直線、箭頭直線，他的移動方式，在繪製了圖形物件後，如果不再指定起始點，那麼，x 軸的位置會平移過去，但 y 軸的位置則維持在原點的位置，這樣說有點抽象，只有請大家試著去畫看看才能體會了，但最好就是指定好各個圖形物件的起始位置，才不容易搞錯。

1. \put(啓始座標){圖形物件}
   將圖形物件置於啓始座標。這個圖形物件也就是 picture 環境的繪圖指令，也可以是一般的文字敘述，如果是文字，那麼會依 LaTeX 的排版方式來顯現，在這篇文章裡頭，有時也會稱爲「圖文物件」。

2. \line(向量座標){長度}
   以參考原點和向量座標所構成的斜率畫指定長度的直線。不過，LaTeX 的這個畫直線的指令，有其限制：
   (a) 兩座標值必需互質。
   (b) 座標值要在 -6 和 $+6$ 之間的整數。
   (b) 座標值必需爲整數。
   所以，實際上只能畫出 25 種斜率的直線，超過這個限制的直線，只能使用較複雜的 \qbezier 指令來畫出來。

3. \vector(向量座標){長度}
   和 \line 指令的作用及使用方法相同，但限制更嚴格，座標值要在 -4 和 $+4$ 之間，和 \line 不同的是向量方向的那一端會多了個箭頭符號。直線和箭頭直線，他們的參考起啓點如果沒有另行指定，那 x 軸的值是會連續的，也就是說畫了一條直線後，再接著畫另一條直線，那他的 x 軸起啓點是由前一條直線的終點開始，但 y 軸的值則沒有這個特性。

4. \circle{半徑}
   畫圓指令。請注意，如沒有使用 \put，則圓心是在原點。如果是使用 \circle*{半徑} 則是實心的圓，常常用來畫某個粗點。由於圓是以圓心爲參考點，並沒有方向性，所以，並不像直線一樣，x 軸的位置會平移，仍然會以原來的原點爲圓心。但如果前面有直線，那麼圓心的位置會受前一個直線影響，也就是說圓心的 x 軸位置會是前一條直線的終點的位置，當然，y 軸的位置不會受其影響，正圓及橢圓都是一樣。

5. \oval{寬,高}[顯示部份]
   畫橢圓。「顯示部份」指的是要畫上半部（t），或是畫下半部（b），或是畫左下半部（bl），依此類推。不管是否完全畫出，圓心仍然是位在完整畫出時的圓心位置。

6. \qbezier[曲線總點數]{起點座標}{控制點座標}{終點座標}
   畫 quadratic Bézier 曲線。其中的「曲線總點數」，代表整條曲線的總點數，有指定的話，曲線會變成虛線，不指定的話是實線，至於什麼是控制點（control point），他可以控制曲線的弧度，可由數學運算計算出來。有興趣的話，請參考：

http://www.ursoswald.ch/metapost/tutorial/BezierDoc/BezierDoc.pdf

7. \thicklines
   指定用較粗的線條，無需接任何參數。使用 \thinklines 可還原爲預設值。

8. \thinklines
   指定用較細的線條，這是預設的線條組細大小，亦無需接任何參數。

9. \linethickness{粗細單位}
   指定線條的預設粗細。

10. \framebox(寬, 高)[框內位置]{圖文物件}
    畫實線框。所謂的「框內位置」可有 t, b, l, r, s，表示圖文物件置放於方框中的位置。

11. \dashbox{虛線線段長度}(寬, 高)[框內位置]{圖文物件}
    畫虛線框。

9.3.3 簡化座標位置

選定一個座標定點，我們可以使用 \put(座標) 的方式來指定，但如果是有規律性重複出現的圖形物件，這樣一個一個指定，不僅很煩，而且也較耗記憶體，計算也會比較慢。這時可以使用 \multiput 指令，他的語法如下：

```
\multiput(起啓座標)(座標遞增值){重複次數}{圖形物件}
```

這裡舉一個例子，畫一個有格子的座標系：

```
% example25.tex
\documentclass{article}
\usepackage{epic}
\parindent=0pt
\begin{document}
\unitlength=1mm
\begin{picture}(80, 60)
\multiput(5, 0)(5, 0){15}{\line(0, 1){60}} % 畫 15 條直線，每隔 5mm 一條
\multiput(0, 5)(0, 5){11}{\line(1, 0){80}} % 畫 11 條橫線，每隔 5mm 一條
\thicklines
\put(0, 0){\vector(0, 1){60}} % 畫 y 軸
\put(0, 0){\vector(1, 0){80}} % 畫 x 軸
\put(0, 0){\circle*{1}}       % 畫圓點，實心粗點
\put(-5, -5){$O(0, 0)$}       % 標上原點的座標
```

```
\put(-5, 60){$y$}              % 標上 y 軸字樣
\put(80, -5){$x$}              % 標上 x 軸字樣
\end{picture}
\end{document}
```

我們來看看這個 multiput 到底做了些什麼事：

```
\multiput(5, 0)(5, 0){15}{\line(0, 1){60}}
```

第一個座標 (5, 0) 是啓始座標，接著的 (5, 0) 是遞增值，也就是說 (5, 0), (10, 0), (15, 0)…(75, 0) 會畫後面所接的圖形物件，也就是畫長度爲 60mm 的垂直線 15 次。由於我們在 x 軸及 y 軸是另外畫帶有箭頭的直線，因此，縱橫軸的部份可以少畫一條直線。x 軸爲 0 的 \line 就是在畫垂直線，y 軸爲 0 的則是在畫水平線。試想想看，這些線條如果要由 \put 指令一個一個畫上去的話，會有多煩！

畫這些除了練習外，主要是給初接觸 picture 環境的朋友一個建議，那就是把方格子畫上去，有利於繪圖時找位置，等眞正要畫的圖畫好了，再把方格子拿掉。編譯好的例子如下：

http://edt1023.sayya.org/tex/latex123/example25.tex
http://edt1023.sayya.org/tex/latex123/example25.pdf

另外一個簡化座標的方式，就是使用 \shortstack 指令，他的語法如下：

```
\shortstack[位置]{圖文物件}
```

這會像疊羅漢一樣的把「圖文物件」疊在一個欄位內，和疊羅漢不同的是，後進的疊在最下面，先進的會被往上堆高，底部的基準線是固定的，高度則是往上增高，各圖文物件由換行符號來換行，也就是說可以由換行符號來決定他們之間的間隔。當然，這要自行注意他的高度，否則會和其上的其他內容重疊。「位置」可爲 l, r, c 之一，是指居中，或靠這個欄位的左右邊的意思。

\shortstack 的一個特殊的運用，就是在座標圖上標註縱軸的文字，但這通常是用在中文，因爲，一般的慣例，縱軸的說明文，英文的話是沿縱軸由下往上寫，中文的話是由上往下寫。我們把 example25 標上中文，實際要加入的內容爲：

```
  ...
\put(-7, 20){\shortstack{這\\[-2pt]裡\\[-2pt]是\\[-2pt]縱\\[-2pt]軸}}
\put(30, -6){這裡是橫軸}
  ...
```

從這裡，我們也可以發現，把格子畫出來，對於繪圖或加入說明文字時的定位非常方便。請注意，這個例子使用了 CJK 環境，要使用 bg5latex 來編譯。編譯好的例子如下：

http://edt1023.sayya.org/tex/latex123/example26.tex
http://edt1023.sayya.org/tex/latex123/example26.pdf

當然，這樣一來，字間距也得手動去調整了，理想的話是應該將中文字旋轉才比較能符合原來的字間距，意即，橫排時的字間距和行間距，在直排的時候，兩者要互換過來，但是這樣一來，會造成中文是沿著縱軸往上寫的情形，這就不符合慣例了，但這剛好常常用在中英文混合的說明文場合，中英文混合時，是按英文的慣例，沿著縱軸由下往上寫，我們將 example26 修改一下：

```
  ...
\put(-7, 20){\rotatebox{90}{這裡是 $y$ $axis$}}
\put(30, -6){這裡是 $x$ $axis$}
  ...
```

請注意，數學式子中的額外空白通常會被忽略。編譯好的例子如下：

http://edt1023.sayya.org/tex/latex123/example27.tex
http://edt1023.sayya.org/tex/latex123/example27.pdf

如何恰當的使用，就請大家視需要去調整、運用了。我們甚至可以更進一步的把各別的中文字去分別旋轉後再排上去，而且，通常圖表的說明文字會比正文小一號，就請大家動手練習一下囉！這個 \rotatebox 指令，我們還沒有學到，會在第 9.6.4 小節，頁 122 裡說明。

請將以上所談到的指令，一一去試著畫幾次，大概就能體會出 picture 環境如何畫圖了。

9.3.4 epic 巨集延伸的指令與環境

以下是 epic 所擴充的指令，和 picture 環境配合的話，會使繪圖更得心應手。

1. \multiputlist(起啓座標)(座標遞增值)[tbrl]{物件1, 物件2·····}
   \multiput 是針對同一個圖形物件按規律性來置放，這個指令則是針對，不同的圖物形物件按規律性來置放。他是把所有的物件置放在一個 box 中去排列，因此會有 tbrl 的置放位置的選項參數。

2. \matrixput(起啓座標)(遞增值1){次數1}(遞增值2){次數2}{圖形物件}
   這是 \multiput 的兩種置放規律版本，可指定兩種規律來置放同一圖形物件。

3. \grid(寬, 高)(橫間隔,直間隔)[標註縱橫軸座標起啓值]
   畫方格的指令。我們在前面有使用 \multiput 來畫方格的例子，使用這個 \grid 將

更爲方便、簡潔。選項的部份，就是在方格外圍標註他的座標值，通常使用 (0, 0) 即可。

4. \picsquare{圖形物件}
   這只是讓圓形、橢圓形，會自動產生一個圓心黑點。

5. \dottedline[點的形式]{點間距}(座標1)(座標2)⋯⋯(座標n)
   畫各種不同形式的虛線。會在各座標間連成虛線。點的形式可以使用其他的符號代替。

6. \dashline[延展值]{各點線長}[dash 點間距](座標1)(座標2)⋯⋯(座標n)
   畫各種不同形式的 dash 線。會在座標間連成 dash 線。延展值從 −100 至無限大，可調整各 dash 的間距。「dash 點間距」只是在調整構成 dash 本身的點的間距，非各 dash 間距。

7. \drawline[延展值](座標1)(座標2)⋯⋯(座標n)
   這是加強型的畫線指令，可以將各座標間連成一線。延展值與 \dashline 同，負數值會造成類似 dash 線的效果。

8. \putfile{檔名}{繪圖指令}
   這是引用外來數據資料來繪圖。也就是可以將 XY 座標值數據資料另存放在外部檔案來引用進來，其餘的功能和 \put 一樣。

更詳細的 epic 說明及其實例，可參考 Sunil Podar 所寫的文件 *Enhancements to the Picture Environment of LaTeX*：

http://www.ntg.nl/doc/podar/picman.pdf

9.4 PSTricks 及 PDFTricks 巨集套件

PSTricks 插入了 PostScript 的繪圖指令給 dvips 去處理，所以繪圖功能當然比 picture 環境強大，但有一個缺點是 pdflatex/dvipdfm[x] 編譯時會出問題，得由 latex/dvips/ps2pdf 的方式來製作 pdf 檔，或者另外繪製後，利用 graphicx 巨集來引入獨立的圖檔。

所以，如果是要製作 pdf 檔案的話，那麼處理上會比較不方便，而且，如果又是中文的話，那麼所製作出來的中文 pdf 檔，以目前的 PDFLaTeX 的話，對於中文將會沒有

copy&paste&search 的功能。PDFTricks 雖然可以讓 pdflatex 使用 PSTricks，但他是利用了 TEX 系統引用作業系統 shell 的功能，這在非 Unix-like 的作業系統可能會出問題，而且他和 latex/dvipdfm[x] 的相容性並不是很好。如果製作的是英文文件，那倒是沒有關係，反而可以使用 PSTricks 的強大繪圖功能。

9.4.1 PSTricks 的組成巨集

PSTricks 除了主要的 pstricks 巨集外，另外還有其他的巨集，專門處理各種不同的特殊繪圖。例如：pst-node 及 ps-tree 可用來畫樹狀圖，pst-grad 可以表現漸層顏色，pst-poly 可以繪製多種的多邊形，pst-gr3d 可以畫 3D 格子圖。通常，一般用途的話，就是引用 pstricks，而他的繪圖環境是包在 pspicture 環境裡頭的，他的預設單位是 1cm，和 picture 環境的 1pt 不同：

```
  ...
\usepackage{pstricks}
  ...
\begin{document}
  ...
\begin{pspicture}(左下角座標)(右上角座標)
  這裡繪圖
\end{pspicture}
  ...
```

他的其他專用巨集並不會自動引用 pstricks，所以要自行引入後，再引用專用的巨集，另外，如果要使用顏色的話，由於 pstricks 對顏色的定義和 LATEX 的巨集會有不相容的情形，因此，我們要引用 David Carlisle 另外寫的 pstcol 這個巨集來修正他。pstcol 會自動引用 color 及 pstricks 這兩個巨集，因此，除非要加入選項參數，引用了 pstcol 就不必引用 color 及 pstricks 了。pstcol 也可以引用和 color 一樣的參數，例如：

```
  ...
\usepackage[usenames,dvipsnames]{pstcol}
  ...
```

由於篇幅的關係，這裡不準備詳細介紹 PSTricks 的繪圖指令，但 TUGIndia(Indian TEX Users Group) 已經完成了相當不錯的 PSTricks 入門教材，請大家不要錯過：

http://sarovar.org/projects/pstricks/
http://www.tug.org.in/tutorials.html

9.4.2 PDFTricks 的使用

這個套件一般系統是沒有安裝的，請至 CTAN 下載，下載後解開，如果是在 Unix-like 系統，那執行 make 就可以了，否則請將 pdftricks.sty 拷貝至：

```
$TEXMF/usr/share/texmf/tex/latex/pstricks  或
$TEXMF/usr/share/texmf/tex/latex/pdftricks  請自行建立目錄
執行 texhash 或 mktexlsr
```

另外，將 pst2pdf 這個可執行檔，拷貝至執行路徑可及之處就行了。另請注意他的檔案格式，如果是在 Unix-like 系統，則要把他的檔案格式改爲 Un*x 系統的規格。

PDFTricks 主要是把 PSTricks 巨集的引用，由 psinputs 環境包起來，而 pspicture 環境，則另外由 pdfdisplay 環境包起來，這樣就可以由 pdflatex 來編譯了。例如：

```
\documentclass{article}
\usepackage{pdftricks}
\begin{psinputs}
 \usepackage[usenames,dvipsnames]{pstcol}
 所有 pstricks 的巨集引用，都要被 psinputs 環境包起來
\end{psinputs}
  ...
\begin{document}
  ...
\begin{pdfdisplay}
\begin{pspicture}
  這裡依正常 pstricks 的指令繪圖
\end{pspicture}
\end{pdfdisplay}
  ...
\end{document}
```

這樣就可以由 pdflatex 直接編譯了。要非常注意的是，執行 pdflatex 時，後面要加上 -shell-escape 參數，這個功能，一般的 TEX 系統是關閉的，要打開來讓他可以由 pdflatex 執行當中，停下來等待外部指令的執行完畢後再繼續原來未完成的工作。不過，這個方式不保證能在 Windows 系統中正確執行。

這裡舉一個簡單的例子，同時說明如何使用 PSTricks、PDFTricks 及如何引入外部 gnuplot 製作的 picture 環境文稿的例子：

http://edt1023.sayya.org/tex/latex123/test-pstricks.pdf
http://edt1023.sayya.org/tex/latex123/test-pdftricks.pdf
http://edt1023.sayya.org/tex/latex123/test-pstricks.tex
http://edt1023.sayya.org/tex/latex123/test-pdftricks.tex
http://edt1023.sayya.org/tex/latex123/gnuplot-label.dem

最後一個 tar ball 是原始文稿。latex/dvips/ps2pdf 的編譯的方式如下：

```
gnuplot gnuplot-label.dem  => 產生 gp-test.tex picture 環境文稿
latex test-pstricks.tex
dvips -o test-pstricks.ps test-pstricks.dvi
ps2pdf test-pstricks.ps
```

pdflatex 的編譯方式如下，請不要忘了 -shell-escape 參數：

```
[gnuplot gnuplot-label.dem]  如果前面已製作好，這裡就不必執行了
pdflatex -shell-escape test-pdftricks.tex
```

關於這些在 LaTeX 繪圖的例子另請參考 Urs Oswald 所舉的例子及其說明：

http://www.ursoswald.ch/LaTeXGraphics/overview/overview.html
http://www.ursoswald.ch/LaTeXGraphics/overview/latexgraphics.pdf

從這些例子可以發現，picture 環境及 PSTricks 巨集的確是可以拿來繪製精美的圖形的。

9.5 METAPOST 使用簡介

METAPOST 除了可以直接繪圖，還可以做一些數學運算後把結果圖形化，所以不必如 picture 環境般的一筆一筆的畫上去，只要能整理出規律出來，就可以使用數學函數的方式來繪圖。在使用上和 picture 環境一樣，要有座標系的觀念。

9.5.1 如何編譯 METAPOST 圖檔文稿？

通常 METAPOST 文稿我們以.mp 爲他的延伸檔名，以便和其他檔案辨別，這個文稿使用任何一種編輯器編輯即可。在英文環境，編譯 METAPOST 文稿的方法可有兩種，中文環境的話，我們會在第 9.5.6 小節討論：

1. 使用 mpost
   這是最正統的方式。底下以 some.mp 爲例，這裡假設裡頭只有一個圖檔，他的編號是 1：

   ```
   mpost some.mp    % 產生 some.1 及 some.log
   epstopdf some.1  % 產生 some.pdf 圖檔
   ```

   some.1 就可以直接引入文稿中了。

2. 使用 mptopdf

   這會產生 pdf 格式的結果，以便給 pdflatex 來引用。其實這只是間接使用了 mpost，主要的編譯程式仍然是 mpost，只不過另外會自動去引用外部程式處理轉換成 pdf 格式的步驟而已。另外，他對字型的使用有做特別處理，因此如果圖檔裡頭有使用文字的話，可能 mptopdf 會處理的比較好。

   ```
   mptopdf some.mp  % 產生 some.1 及 some-1.pdf
   ```

   這樣也可以產生 some.1，及多出一個 some-1.pdf。但 mptopdf 所產生的 some.1 是眞正的 eps 檔，可以直接由 gv/gsview 來閱覽，傳統 mpost 所產生的 some.1 則是使用 METAPOST 的字型表示法，所以 gv/gview 會無法正常閱覽。不過，引進 LaTeX 文稿則沒有什麼差異。

3. 編譯結果的驗證方法

   由於 mpost 編譯後的 eps 圖檔，gv/gsview 並不一定能閱覽，所以，我們要驗證他的編譯結果，除了引入文稿中外，還有一個標準的驗證方法：

   ```
   tex mproof some.1 ... some.n  % 產生 mproof.dvi
   dvips mproof.dvi              % 產生 mproof.ps
   ```

   可以接受多個圖檔，看原來的文稿中有幾個圖檔，後面就接幾個圖檔，他會集中所有圖檔顯示在 mproof.dvi 當中，再利用 dvips/dvipdfm[x] 就可以產生 ps/pdf 檔來閱覽，他會加入各個圖檔的檔名及編號。

9.5.2 METAPOST 文稿的基本結構

在談到 METAPOST 文稿的結構前，我們先提一下 METAPOST 使用的度量單位，如果沒有標明的話，他預設是使用 big point(bp)，也就是 PostScript 規格中所使用的單位，這和 TeX 本身是使用 printer point(pt) 不一樣。這裡我們所談的，主要是用於繪圖，至於他的巨集功能，這裡就不討論了。他的文稿結構如下：

```
% 和 TeX 一樣，註解是使用百分號
beginfig(n);
這裡寫 METAPOST 的繪圖指令敘述，每個指令敘述以 ; 結尾
endfig;
bye;   % 這一行可以不必 ;。這是通知 mpost 程式跳出，結束繪圖，end 亦可。
```

其中的 beginfig(n) 的 n 可以是任何一個數目字，代表經 mpost 編譯過後所輸出的 eps 圖檔的延伸檔名，如果一次要處理多數圖檔，只要使用多個 beginfig…endfig 就可以

了，但其內的數字則不可相同，以免圖檔被覆蓋。METAPOST 的指令和 TEX/LATEX 不同的是前面沒有 \ 來起頭，而且每個敘述最後一定要加個; 來結束，否則編譯會出錯，例外是 beginfig、endfig 及 bye/end 這些指示可以不必使用; 來結束。

9.5.3 METAPOST 的九種基本資料型態

這些資料型態可以儲存固定的繪圖指令組或者是數學運算式，這樣的好處是，可以把一組複雜但又常重複的繪圖指令集或數學運算，宣告成幾個變數來代表，往後要用到時，就使用變數即可，可以讓 METAPOST 的程式碼寫起來更簡潔。

1. numeric

   數值，他的大小要在 -4096 與 4096 之間，中間的計算值也不能超過這個限度的 8 倍。一般宣告的例子如下：

   ```
   numeric a, b, c, d; % 宣告 a,b,c,d 四個變數爲數值型態
   a := 9;
   b := 3*a**2;
   c := a++b;
   d := c+-+b;
   show a, b, c, d;     % 把計算結果顯示出來
   bye
   ```

   其中 3*a**2 代表 $(3a)^2$，a++b 代表 $\sqrt{a^2+b^2}$，c+-+b 代表 $\sqrt{a^2-b^2}$。METAPOST 不支援科學計數表示法。這可以存成一個檔案，經由 mpost 編譯後會顯示計算結果，也可以由以下的方式來直接輸入計算：

   ```
   edt1023:~$ mpost
   This is MetaPost, Version 0.641 (Web2C 7.4.5)
   **\relax
   * => 變成一個星號時，就可以輸入各行敘述了
     ...
   ```

   這裡要注意的是，numeric 型態不一定要事先宣告，可以直接使用，但宣告的用意是在消除前面的指定，也就是說，一經宣告就會消除以前所指定的值。所以，直接使用的變數，METAPOST 會把他當做是 numeric 型態。

2. pair

   這是指平面座標值，各 pair 間可做四則運算。例如，我們指定[3]：

[3]METAPOST 的指定（assignment）運算子是:=，一般的等號才是等於，但第一次指定時兩個都可以代表指定運算，其後更改指定的話就一定要使用:=。

```
beginfig(1);
u := 1mm;          % 指定單位
pair a, b, c, d;  % 宣告四個 pair 形態的變數
a := (0, 0); b := (30u, 0); c := (30u, 30u); d := (0, 30u);
draw a--b--c--d--a;
endfig;
bye
```

這會繪製 30mm 長寬的正方形。也可以使用矩陣的方式來宣告，例如：

```
pair a[];   % 這等於宣告了 a0, a1, a2, a3...
pair a[][]; % 這樣宣告也可以，代表 a00, a01...,a12, a13...
```

3. path

   有方向性的 pair 都可以是 path。例如：直線、曲線。

4. color

   顏色，由 rgb 值所組成，例如：(0, 0, 0) 是黑色，(1, 1, 1) 是白色。預先定義好的顏色有:

顏色	rgb 值
black	(0, 0, 0)
white	(1, 1, 1)
red	(1, 0, 0)
green	(0, 1, 0)
blue	(0, 0, 1)

   在 METAPOST 使用顏色比在 LaTeX 裡頭方便多了，各種顏色的表示，除了原來的 rgb 值的表示法外，也可以在各種預先定義好了的顏色加上其深淺度，例如：

```
withcolor .6red;
withcolor .5white;
color A;           % 宣告 A 爲 color 型態的變數
A := (.3,.8,.2);  % 指定 A 的顏色
withcolor A;       % 使用顏色 A
```

5. string

   字串，在 METAPOST 中，字串可能兩種情形，一種是單純的字串，以" 框起來即可，另外一種是由 btex 及 etex 包住的字串，這些字串，METAPOST 並不去處理，而是交給 tex 去處理。

   通常，我們常常需要把計算出來的數值結果標示在圖上，這時要把他轉換成字串的型態才能標示上去，我們可以使用 decimal() 函數來轉換。因爲我們不能把變數值放在標註指令內，否則他將無法進行運算。例如：

```
u := 1cm;
for i=0 upto 10:
  label.top(decimal(i/10), ((i+1/2)*u,1u));
endfor;
```

這時，變數尚未計算出來，因此不能當做 label.top() 的參數，需要在結果計算出來後，馬上進行轉換成爲字串型態。

6. boolean
   布林值。

7. picture
   任何可以由 METAPOST 繪出來的圖形，皆可宣告成 picture 變數。

8. pen
   畫筆。

9. transform
   轉換。

9.5.4 METAPOST 常用的指令及函數

METAPOST 的指令，基本上可分成兩大類，繪圖指令（picture command）及標籤指令（label command）。繪圖指令用於繪圖，例如 draw、fill 等等；標籤指令則用於標註文字，例如 label、dotlabel 等等。繪圖指令後面可以再接附加指令（addto command）及修整指令（clip command），例如 withcolor、withpen、clip...to 等等，附加指令及修整指令都是可以省略的；標籤指令接的則是要標註的文字及其位置，這些則不能省略。整體的語法結構算是很緊密的結合在一起，所以這裡就不完整列出來了，但實際使用則還算口語化，從實際例子去熟悉用法，可能會比較容易進入狀況。

1. 畫線（draw--）
   由 draw 指令來開始畫圖，各座標值（pair）由兩個 hyphen（就是鍵盤上的減號）-- 來連接各個 pair，這就會構成直線，最後連接至 cycle 的話，就會形成封閉區域的的各種形狀的「框」。有附加指令的情形，例如：

```
pair a, b, c;
a := (0, 0); b := (1cm, 0); c := (0, 1cm);
draw a--b--c--cycle withpen pencircle scaled 1bp
  withcolor .8white dashed withdots;
```

棕色的部份就是附加指令，他的意思是設定線條的粗細爲 1bp（預設值是 0.5bp），使用的顏色是灰色，並且是虛線，附加指令可以省略全部或一部份的敘述。如果附加指令的部份會使用二次以上的話，也可以將他固定下來，這樣就不必每 draw 時都要去指定，例如：

```
pair a, b, c;
a := (0, 0); b := (1cm, 0); c := (0, 1cm);
pickup pencircle scaled 1bp;                          % 固定線條粗細
draw a--b--c--cycle withcolor .8white dashed withdots;
```

2. 設定 draw 預設值（drawoptions()）
   如果我想更進一步的設定 draw 的預設值，我們可以使用 drawoptions() 函式，例如前面的例子，可以設成：

```
pair a, b, c;
a := (0, 0); b := (1cm, 0); c := (0, 1cm);
drawoptions(withpen pencircle scaled 1pt dashed withdots withcolor .8white);
draw a--b--c--cycle;
```

   要注意的是，如果有標註文字，這會連圖上的標註文字也使用所指定的顏色。

3. 畫弧線（draw..）
   和畫直線完全相同，只是把兩個 hyphen 換成兩個句點.. 來連接各個 pair。

4. 畫箭號（drawarrow, drawdblarrrow）
   和 draw 指令的用法一樣，只是改成 drawarrow（單箭號）及 drawdblarrow（雙箭號）。

5. 畫虛線
   在畫線指令後附加上 dashed evenly 或 dashed withdots 即可。這是屬於附加指令的敘述。我們也可以由 dashpattern() 來自行定義虛線的形式，例如：

```
pair a, b, c;
picture dptn;
a := (0, 0); b := (1cm, 0); c := (0, 1cm);
pickup pencircle scaled 1bp;                          % 固定線條粗細
dptn := dashpattern(on 6bp off 2bp on 2bp off 2bp); % 指定 dash 的形式
draw a--b--c--cycle withcolor .8white dashed dptn;
```

   dashpattern 的部份，on 表示會顯示出來的線段，off 表示不顯示出來的空白。預設指定的形式是：

```
evenly   := dashpattern(on 3 off 3);
withdots := dashpattern(off 2.5 on 0 off 2.5);
```

這裡的 evenly 及 withdots 就是屬於 picture 型態的變數。

6. 填色

使用 fill 指令來代替前面的畫線指令，後面要加個 withcolor 的指令來指定顏色，否則會填入黑色。要注意的是，fill 必須是封閉區域才有意義，因此，連接 pair 時最後一定要使用 cycle 把區域封閉起來。

要注意的是，他僅僅著色，不畫線框，要畫線框要另外由 draw 指令來畫。或者改用 filldraw 指令，這樣可以既畫框又填色，只不過，這樣一來都是同一種顏色了，如果框線和所填的顏色要不一樣，就得經過 draw 及 fill 兩道手續。

7. 畫圓

draw 後附加個 fullcircle，這畫的是正圓。也可以畫其他種類的圓：

圓的種類	指令
完整正圓	fullcircle
四分之圓	quartercircle
半圓	halfcircle

```
draw fullcircle scaled 1cm  % 圓心 (0, 0)，直徑 1cm 的圓
draw fullcircle scaled 1cm shift (x, y)  % 圓心平移至 (x, y)
draw fullcircle xscaled a yscaled b  % 長短軸各爲 a, b 的橢圓
```

前面曾出現的 with... 附加指令都可以使用。其他如半圓、四分之一圓的使用方法相同。

8. 畫方框（unitsquare）

前面曾學過由畫線連起來可以畫正方形，但畫方框有更簡單的方式，那就是使用 unitsquare 附加指令，例如：

```
draw unitsquare scaled 1cm;  % 左下角是 (0, 0)，寬高 1cm 的正方形
draw unitsquare scaled 1cm rotated 30;    % 將正方形逆時針旋轉 30 度
draw unitsquare xscaled 2cm yscaled 1cm;  % 寬 2cm，高 1cm 的矩形
```

而且，也可以使用 shift(x, y) 來平移左下角座標的位置。

9. 標註文字（label, dotlabel）

我們先把標註文字時的位置弄清楚，再來看怎麼標註上去：

top（正上方） ulft（左上角） urt（右上角）
lft（左） • rt（右） •
bot（正上方） llft（左下角） lrt（右下角）

由其中的英文字縮寫，應不難明白他們的意思。中央的黑點，代表標註文字時所給的座標位置。所以，我們的文字可以標示在一個定點的八個方位。例如：

```
label.bot("O(0,0)", (0,0));     % 在原點正下方標示 O(0,0) 字樣
dotlabel.bot("O(0,0)", (0,0)); % 和 label 一樣，但會有個粗黑點
label.bot(btex $O(0,0)$ etex, (0,0));   % O(0,0) 字樣經由 \TeX\ 排版
dotlabel.bot(btex $O(0,0)$ etex, (0,0));
```

由" 包圍的，他的文字是由 METAPOST 自行處理，就是一般的文字表現，但由 btex…etex 包圍的，則會交給 TEX 去排版，例如這裡進入數學模式，就會以數學斜體來排版。

這裡舉一個簡單的例子，rgb 三原色及灰階的漸層演色表，順便使用了我們沒有提到的 for loop[4] 及調整字型大小的方法（這些會特別用底線標示出來）。

```
% test-mpcolor.mp
beginfig(1);
defaultscale := 12pt/fontsize(defaultfont); % 使用 12 點字
u := 1cm;
path sqr;
sqr := unitsquare scaled u;
for i=0 upto 10:   % for loop 迴圈，請注意這裡是冒號
label.top(decimal(i/10), ((i+1/2)*u,1u));
fill sqr shifted (i*u, 0) withcolor i*0.1red;
fill sqr shifted (i*u, -1.2u) withcolor i*0.1green;
fill sqr shifted (i*u, -2.4u) withcolor i*0.1blue;
fill sqr shifted (i*u, -3.6u) withcolor i*0.1white;
endfor;
label.lft("r", (0,.6u));
label.lft("g", (0,-.6u));
label.lft("b", (0,-2u));
label.lft("gray", (0,-3u));
endfig;
bye
```

由 mptopdf 執行的結果如下：

http://edt1023.sayya.org/tex/latex123/test-mpcolor.mp
http://edt1023.sayya.org/tex/latex123/test-mpcolor-1.pdf

9.5.5 和 LATEX 的配合

METAPOST 主要是用在 TEX 文稿，但由於他可以引入 TEX 巨集，也因此表示可以把 LATEX 的一些指令及巨集引進去，這樣，編譯的時候，該呼叫 tex 的，就會改呼叫

[4] METAPOST 有完整的條件判斷式及迴圈，但這已經屬於程式設計的範圍，因此這篇文章裡頭並沒有詳細說明，只有舉一個 for loop 的典型例子，有需要深入時請自行參考文章後面的參考資料。

latex。當然，這也表示，我們可以使用 CJK 環境來書寫中文了。

通常會要呼叫 tex 及 latex 的場合，就是要使用他們的排版功能，主要是用於排文數字，可以加入一般的文字，經過 TEX/LATEX 排版後的漂亮結果，也可以是高品質的數學式子。

1. 加入文數字
   只要是包在 btex…etex 裡頭的文數字都會交給 TEX 去排版。當然，僅僅使用 " 包住也是可以，但就不經過 TEX 處理了。

2. LATEX 的情形
   要使用 LATEX 要經過一些處理，因爲 METAPOST 預設是把排版交給 tex 去編譯的，這時要把所需要用到的 LATEX 巨集及指令包在 verbatimtex…etex 當中，例如：

```
verbatimtex
%&latex          % 指示由 latex 編譯，而不是預設的 tex
\documentclass{article}
\usepackage{some,packages}
\begin{document}
etex
  ...
verbatimtex      % 以下這段通常不必，但如果有引入其他環境時則不可省略
\end{document}   % 這段放在文稿最後即可
etex
```

這樣一來，包在 btex…etex 之間的文字就可以使用 LATEX 指令去排版，而 mpost 也會交給 latex 去處理文字的部份。

9.5.6 在 METAPOST 中使用中文

METAPOSTindexmetapost@METAPOST 文稿中，重點當然是畫圖，但是，有時也是要加入一些文字，這在英文是很方便，但在我們的 Big-5 中文就很頭大了，這裡我們使用了一個小工具 b5mp.pl，這個工具主要是引用王佑中[7] 先生的 clatex 中的一個函式，把會出問題的 Big-5 中文處理好。然後，我們再加入必要的 LATEX 及 CJK 環境的結構，最後呼叫 mpost 來編譯他，這個小工具是由 perl 寫的，可以在此下載：

http://edt1023.sayya.org/tex/latex123/b5mp.pl

他的使用方法很簡單，把他當成是 mpost 即可，使不使用.mp 延伸檔名都沒有關係。如果 fontname 沒有指定，那麼預設是 aming：

```
b5mp.pl [fontname] your[.mp] => 這樣會產生 your.1（視圖檔內的編號而定）
perldoc b5mp.pl              => POD 格式的 b5mp.pl 使用說明
```

這個 your.1 就可以被引進 LaTeX 文稿裡頭去了。由於這個編譯出來的 eps 檔，含有中文字型資訊，所以 gv 及 gsview 都會無法閱覽，epstopdf 也是無法處理，我們可以使用以下的方法來「預視」：

```
tex mproof your.1      => 這會產生 mproof.dvi
dvips mproof.dvi       => 產生 mproof.ps
或
dvipdfm[x] mproof.dvi  => 產生 mproof.pdf
```

這樣就可以去預視了。當然，你的 LaTeX 系統要安裝好 CJK 套件，否則還是會認不得這些中文字型資訊的。mptopdf 除非另做些修改，否則在中文的場合會無法正常使用。我們這裡就綜合舉一個例子：

```
% test-yi.mp
beginfig(1)
u:=3cm;
path p;
p=(0,1u)..(1u,0)...(0,-1u);
fill p{dir(157)}..(0,0){dir(23)}..{dir(157)}cycle;
draw p..(-1u,0)..cycle;
fill (0,-.6u)..(0.1u,-.5u)..(0,-.4u)..(-.1u,-.5u)..cycle withcolor white;
fill (0,.6u)..(.1u,.5u)..(0,.4u)..(-.1u,.5u)..cycle;
label.bot(btex \Large 仿太極陰陽魚圖 etex,(0,-1.2u));
endfig;
beginfig(2)
a=.7in; b=0.5in;
z0=(0,0); z1=(a,0); z2=(0,b);
z0=.5[z1,z3]=.5[z2,z4];
draw z1..z2..z3..z4..cycle;
drawarrow z0..z1;
drawarrow z0..z2;
label.top(btex \small 橫軸 $x$ etex, .5[z0,z1]);
label.lft(btex \small 縱軸 $y$ etex, .5[z0,z2]);
label.bot(btex \Large 許功蓋測試 etex,(0,-.7u));
endfig;
end;
```

這裡頭有兩張圖，都有使用到中文，所以我們要使用 b5mp.pl 來代替 mpost 來編譯：

```
b5mp.pl akai test-yi  % 使用 akai 字型，並產生 test-yi.1 及 test-yi.2
tex mproof test-yi.1 test-yi.2  % 產生 mproof.dvi
```

```
dvips mproof.dvi                % 產生 mproof.ps
dvipdfmx mproof.dvi             % 產生 mproof.pdf
```

編譯好的例子如下：

http://edt1023.sayya.org/tex/latex123/test-yi.mp
http://edt1023.sayya.org/tex/latex123/mproof.pdf

現在我們再來看看 LaTeX 文稿中要如何引用：

```
% example28.tex
\documentclass{article}
\usepackage{graphicx,CJK,mflogo}
\parindent=0pt
\ifx\pdfoutput\undefined
  \DeclareGraphicsRule{*}{eps}{*}{}
\else
  \DeclareGraphicsRule{*}{mps}{*}{}
\fi
\begin{document}
\begin{CJK}{Bg5}{hwmm}
這是一個 \MP{} 文稿中使用中文的例子，經由 {\ttfamily b5mp.pl} 編譯後引入
\LaTeX\ 文稿當中。圖中的「太極陰陽魚」字樣是 \MP{} 文稿中就有，
不是這裡鍵入的。但其實圖檔裡頭並沒有眞正的字型，而是引入文稿，
經 {\ttfamily latex} 編譯後，才「合作」眞正產生的。
\vspace{10ex}
\begin{figure}[h]
\centering
\includegraphics{test-yi.1}
\caption{太極生兩儀}
\end{figure}
\vspace{10ex}
\begin{figure}[h]
\centering
\includegraphics{test-yi.2}
\caption{英文字是數學斜體}
\end{figure}
\end{CJK}
\end{document}
```

編譯好的例子如下：

http://edt1023.sayya.org/tex/latex123/example28.tex
http://edt1023.sayya.org/tex/latex123/example28.pdf

9.5.7 更多的 METAPOST 的實例

這裡只是簡單的介紹了 METAPOST，實際要用的話，複雜一點的圖形，可能會不太足夠，Knuth 教授的 *The METAFONTbook*，這應該最詳細的資料了，其中有些地方會和 METAPOST 不一樣，但整個結構上是相同的，可和系統上就有的 `mpman.ps` 這個 METAPOST 作者 John D. Hobby 寫的使用手冊相互參照就可以清楚不同的地方。

關於 METAPOST 更多的例子，也另請參考 André Heck 所寫的 *Learning METAPOST by Doing* 及 Urs Oswald、Vincent Zoonekynd 所舉的實例及其解說，`examples.zip` 檔是 `metapost.html` 網頁的原始碼，收錄在 CTAN 中：

```
http://remote.science.uva.nl/~heck/Courses/mptut.pdf
http://www.ursoswald.ch/metapost/tutorial.pdf
http://tex.loria.fr/prod-graph/zoonekynd/metapost/metapost.html
ftp://ctan.unsw.edu.au/tex-archive/info/metapost/examples.zip
http://www.topology.org/tex/conc/mp/mp.zip
```

最後一個 `mp.zip` 是 Alan U. Kennington 所著 *differential geometry reconstructed: a unified systematic framework* 一書中使用 METAPOST 繪圖的實例原始碼。

9.6 圖形的引入

這裡我們使用 graphicx package 來說明，他會自動引入 graphics package，這兩個 package，主要是一些指令的參數用法不同，由於 graphicx 的參數用法彈性較大，而且也和 LaTeX 的一些參數的形式較符合，因此，我們就以 graphicx 來說明，引入巨集時就引用 graphicx 就可以了。這一節所說的是外來的圖檔及 METAPOST 圖檔，而非由 `picture` 環境或 PSTricks 巨集所繪製的圖檔。

9.6.1 引入外來圖檔的方法

講繪圖工具講了老半天，到底有了圖檔，要如何引進文稿裡頭呢？就是使用 graphicx package 的 `\includegraphics` 指令：

```
  ...
\usepackage{graphicx}
  ...
\begin{document}
```

```
  ...
\begin{figure}                          % 進入浮動環境
\includegraphics[選項參數]{圖檔名稱}
  ...
\caption{這裡加入圖的標題}              % 圖號會自動編號
\label{這裡加入引用圖檔時的文字標誌} % 一定要在 caption 之後
\end{figure}
  ...
```

這樣就行了，就這麼簡單！不使用浮動環境也是可以的。通常目前的 graphicx 巨集會自動判斷圖檔格式，所以延伸檔名可以不必寫上。但如果引入的圖檔是 METAPOST 所產生的，那要附上延伸檔名，通常是數目字，例如 some-graphic.1，但這在 pdflatex 可能會有誤認檔案格式的情形發生，所以，要讓他知道這是 METAPOST 所產生的 mps 圖檔。因此，如果文稿中有引用 METAPOST 的圖檔，最保險的方式是：

```
\documentclass{article}
\usepackage{graphicx}
\ifx\pdfoutput\undefined
  \DeclareGraphicsRule{*}{eps}{*}{}
\else
  \DeclareGraphicsRule{*}{mps}{*}{}
\fi
\begin{document}
  ...
\includegraphics{mpost-image.1}
  ...
\end{document}
```

有一個 ifpdf 巨集，做的也是同樣的事情。這樣的話，使用 pdflatex 編譯時會把認不得的圖檔，當做是 mps；而使用 latex 編譯時，會把認不得的圖檔，當做是 eps。另外一個方法，就是把 *.1 改延伸檔名爲 *.mps，這樣 latex/pdflatex 都會認得他。

也可以在引用 graphicx 時加入檔案格式選項參數，指定圖檔格式，但這裡不建議這麼做，這樣有可能會使用文稿的彈性降低，如果不想改來改去，比較理想的方式是去修改 graphicx 的相關設定檔，但一般使用者恐怕也是搞不清楚這些設定檔要如何編修，因此還是使用條件式的判斷來暫時解決這個問題。

如果圖檔中有 jpeg/png 圖檔，這在 latex/dvips 是無法處理的，可使用 convert 把他轉成 eps 格式。在這種情形下，如果要讓 latex/pdflatex 都能正常運作的話，可能要準備兩種格式（eps/pdf）的圖檔了，然後，圖檔名稱不指定延伸檔名，讓執行程式各取所需。

有時使用圖檔時如果會有無法判斷 BoundingBox 值的情形，在這種情形下，可以使用

dvipdfm 所附的 ebb 這支小工具來產生必要的 BoundingBox 值。

9.6.2 includegraphics 指令的選項參數

我們還沒談到 \includegraphics 選項參數的部份。這個選項參數很多，功能很強大。這些選項參數可以有多個，各選項間以逗點來分隔，他的值的設定是使用等號（請注意，我們這裡談的是 graphicx 巨集，而不是 graphics，這在參數使用上不同）。

1. bb
   設定圖檔的邊界（bounding box），含四個值，每個值以空白隔開。例如 bb=98 98 468 430，這個意思就是左下角的座標是 (98, 98)，而右上角座標是 (468, 430)，這個參考標準是可被印出紙張的左下角爲 (0, 0)。請注意，如果沒有指定單位的話，那預設是 bp。而且，這個設定在 pdflatex 會不被接受，此時請改使用 trim 選項參數。

   通常，這是會加上 clip 參數，作用是在修剪引入的圖檔的四周，但不是很好控制，所以建議圖檔由圖形處理或轉換程式去處理過後再引入會比較好控制。不加 clip 參數，加個星號 \includegraphics* 作用是一樣的。

   一般如果是 eps/ps 檔，可以使用編輯器去修改他的 BoundingBox 值，無需用到這些不好控制的參數，如何抓到座標值呢？使用 gv 或 gsview 把圖檔載入後，將游標置於圖中所要的位置，這些軟體就會顯示所在處的座標，然後就可以依自己需要去修改他了。

2. clip
   修剪圖的四周指定的邊緣。

3. trim
   作用和 bb 一樣，也是四個參數，但這裡指的是要去除的部份長度值，而非相對於左下角的相對座標。這個參數可以用在 pdflatex。例如：

   ```
   \includegraphics[trim=7 7 7 7, clip]{some}
   ```

   這會除去 some 這個圖檔的四周 7bp 的寬度。請注意，圖檔盡量不要加延伸檔名，讓系統自己去判斷，這樣文稿會比較有彈性。

4. angle
   旋轉的角度。旋轉指的是逆時針的方向轉的，除非使用負數的角度。

5. `orgin`
   旋轉的中心點。

6. `width`
   這是指圖形的寬度，會自動伸縮調整，長度亦會等比例調整。

7. `height`
   這是指圖形的高度，會自動伸縮調整，寬度亦會等比例調整。

8. `totalheight`
   這是指圖形的總高度，即 height 再加上 depth 的值。會自動伸縮調整，寬度亦會等比例調整。

9. `scale`
   按一定比例縮放，這沒有單位，這是縮放倍數。

9.6.3 指定圖檔的搜尋路徑

如果圖檔很多，一個比較方便的方法就是在目前工作目錄下，新開一個子目錄來專門置放圖檔，這樣在文件的維護上也會比較好維護。他的語法如下：

```
\graphicspath{{路徑一}{路徑二}{路徑三}...}
\graphicspath{{images/}}  % 縱使只有一個子目錄，也不可省略大括號
\graphicspath{{:images:}} % Mac 系統的表示法
```

LaTeX 系統預設找圖檔的路徑是 TeX 預設會去找的路徑及目前的工作目錄（通常目前的工作目錄會先找）。或者，也可以修改 `TEXINPUTS` 變數的值（會較有效率，也較省記憶體），例如，以 unix-like 系統下的 sh shell 爲例：

```
TEXINPUTS = "images/:" ; export TEXINPUTS
```

這樣會首先搜尋目前工作目錄下的 `images` 這個子目錄，找不到的話，才會去 TeX 預設的搜尋目錄去找，這樣一來就會很有效率，而且會較省記憶體。所以，這個方法比較建議使用。

當然，也可以直接在 `\includegraphics{}` 的參數裡頭就直接把路徑寫進去，但這是最不建議的方法，不管效率或是文稿可攜性都會很差。

9.6.4 圖文的旋轉

我們常常會需要某些圖文在特別的情況下旋轉一下，\rotatebox 這個指令，其實我們前面舉的例子當中就曾使用過，我們現在來看看詳細的使用方法：

\rotatebox[選項參數]{角度}{圖文物件}

角度和 \includegraphics 的 angle 選項參數一樣，但使用方法則簡化了，直接寫上數值即可，當然預設是逆時針方向旋轉。選項參數的部份可以有三個小選項：

1. origin
   設定旋轉中心點的位置，可以使用 lrctbB 或其中兩個的組合，其中 B 代表的是基線（baseline），其他的依其英文字母就可理解他的意義，如 t 是 top，r 是 right，c 是 center。預設的位置是左下角，文字的話，則是左下角的參考點（reference point），旋轉就是以此點所構成的的軸心線來轉的。

2. x, y
   這是以左下角爲原點，直接設座標，來表示 origin 所能表現的更精確中心點位置。

3. units
   設定旋轉的特殊弧度。其中 units=-360，這樣會把預設的逆時針旋轉，變成順時針旋轉。

旋轉不限於簡單的圖文物件，甚至一整個表格、圖形環境都可以拿來轉。要注意的是，編譯成 *.dvi 檔的話，有可能 dvi viewer 會不支援旋轉效果的解讀，此時要把他由 dvips 來轉成 ps 檔，或直接使用 pdflatex 編譯成 pdf 檔，再來預視。

9.6.5 圖文的縮放及延展

1. \scalebox{水平縮放倍數}[垂直縮放倍數]{圖文物件}
   垂直縮放倍數可以省略，省略時代表等於水平縮放倍數。

2. \reflectbox{圖文物件}
   這其實是 \scalebox{-1}[1]{圖文物件} 的意思，會得到鏡射的效果。

3. \resizebox{寬度}{高度}{圖文物件}
   \resizebox*{寬度}{總高度}{圖文物件}

這是在改變原圖文物件的大小。寬高使用! 代替的話，會依另一個值做同比例的改變。如果加個星號，如 `\resizebox*{}{}{}`，他的作用是會把深度（depth，基線以下的稱爲深度，基線以上的稱爲高度 height，兩者之和稱爲總高度，totalheight）也考慮進去，否則只考慮高度，但深度不一定會有，這時 height 就等於 totalheight。基線、深度等術語的意義，請複習一下第 3.3.1 小節，頁 14。

我們在表格及圖形的章節裡花了比重不少的篇幅，主要的用意是想打破 LaTeX 所能使用領域的刻板印象，個人還覺得這些內容實在還不夠，但要硬塞入這篇入門級的文件，已經不恰當，時間、能力允許的話，希望將來有機會另外專文來做更詳細一點的介紹。

第 10 章

數學排版

好啦！這章是 LaTeX 的拿手把戲了。就讓我們就來見識一下 LaTeX 的威力吧！光這一章的內容就可以寫一本厚厚的書了，所以，只能點到爲止，先小酌一番。這一章的內容，他們排版精確性，以 PDF 格式的內容爲準，HTML 格式的內容，僅供參考。

對於較複雜的數學式子，除非是自行定義巨集，否則 LaTeX 內建所提供的排版數學式子的能力可能會有不足，這時可以使用美國數學協會所開發的 $\mathcal{AMS}$-LaTeX 巨集套件，目前所有的 TeX 發行版本應該都會附上，而且也會附上另一套 AMSFonts 巨集套件及其字型。這個套件的使用，這裡並不做詳細的說明，只在必要的時候附帶提及，可以另外參考系統上所附的 `amsldoc.dvi` 文件及 *The LaTeX Companion* 這本書第八章，這個部份網路上可以抓得到，檔名是 `ch8.pdf`，CTAN 有收錄：

http://www.ctan.org/tex-archive/info/companion-rev/ch8.pdf

10.1 進入數學模式（math mode）的方法

我們平常寫文章的模式無法正確處理數學式子間的空間位置，而且要鍵入次方、方根、積分 …… 等等符號，會有困難，因此，所有的數學式子都得進入數學模式來處理。在數學模式下，不僅大部份文字、符號會採用斜體字，而且空間會另做安排，額外的空白會被 LaTeX 忽略，在數學模式中要鍵入一般的正常文字，要退出數學模式，或者由 `\mbox{}` 或 `\textmr{}` 包圍起來才行。

LaTeX 的數學模式有兩種，一種是和內文排列在一起的隨文數式（math inline mode），他是和一般正常文字混在一起排版的；另外一種是獨立的展式數式（math display mode），他會單獨成一行，而且上下會和正常文字有一定的空間來區隔。

10.1.1 隨文數式（math inline mode）

這是在夾雜在一般文章內的數學式子，是隨著整個文章段落一起排版的。

1. `$` 數學式子 `$`
   其實，我們在前面的章節的例子裡，就已經常常在使用了，只是沒有詳細說明。由兩個錢字符號 `$` 所包圍的內容就會進入隨文的數學模式，在一般文字段落內要使用到一些數學式子的話，這是最方便的方法。爲什麼是使用錢字符號？因爲 Knuth 教授認爲數學是很「昂貴」的！眞正文章中要寫錢字符號時，要把他 escape，寫成 `\$`，大概是指，平常不必把錢看得太重的意思吧（這是我猜的）！:-)

2. `\begin{math}` 數學式子 `\end{math}`
   如果數學式子很長，那麼使用環境的方式亦可。但是這個環境和一般的環境不同的是，他不會在上下行區隔出來，而是隨著其他正常文字一起排版的。要非常注意的是，在這個環境的上下行不要留空白行，否則會另起段落排版，那就不是我們所要的隨文數式了。

3. `\(` 數學式子 `\)`
   這是 `\begin{math}` 數學式子 `\end{math}` 省略寫法。

我們來試試看，到底進入數學模式和不進入數學模式會有什麼不同：

```
f(x,y)=3x+4y                  % 不進入數學模式
f(x, y) = 3x + 4y             % 不進入數學模式，空白是有作用的
$f(x,y)=3x+4y$                % 進入數學模式
$f(x, y) = 3x+  4y$           % 數學模式中留不留空白，及留幾個空白，作用都一樣
sin(2x)=-sin x cos x          % 這樣排版出來會慘不忍睹喔！
$\sin (2x) = -\sin x \cos x$
f(x,y) = 3(x+y)y / (2xy-7)    % 這樣排版出來也是會慘不忍睹！
$f(x,y) = 3(x+y)y / (2xy-7)$
```

排版出來的結果會是：

誤	正
f(x,y)=3x+4y	$f(x,y) = 3x + 4y$
f(x, y) = 3x + 4y	$f(x,y) = 3x + 4y$
sin(2x)=-sin x cos x	$\sin(2x) = -\sin x \cos x$
f(x,y) = 3(x+y)y / (2xy-7)	$f(x,y) = 3(x+y)y/(2xy-7)$

可以看得出來，英文字的部份變成斜體字了，而且加號、逗點、等號前後的空白也不一樣。但是函數名則還是使用正常字體，這在後面第 10.1.3 會談到數學模式中的遊戲規則。

10.1.2 展式數式（math display mode）

通常獨立的數學式子，我們不會使用一般文章一樣的做法去換行，而是讓他進入展式數式的數學模式，他會獨立成一行，有需要的話也可以加入編號，以方便在文章中引用。和隨文數式另一個很大的不同是，展示數式會適當的選用較大的數學符號及字體，尤其是較複雜的數學式子的時候。

1. `\begin{displaymath}` 數學式子 `\end{displaymath}`
   這會使數學式子獨立成一行。
2. `\[` 數學式子 `\]`
   這種方式也可以，也比較常用。這兩種的展示數式都不會編號。
3. `\begin{equation}` 數學式子 `\end{equation}`
   這種使用方式，亦會獨立成一行，而且會附上編號。`equation*` 則不附編號。

使用展式數式要注意的是和上下文章不要空出空白行出來，裡頭也不要空出空白行。請不要使用 TEX 裡頭的 `$$` 指令，這在 LATEX 並沒有完整去重定義他，這在某些 LATEX 指令的效果上會沒有作用。

10.1.3 在數學模式中的一些遊戲規則

在數學模式中，由於一些空間的安排和一般文章段落不一樣，因此在編輯文稿時，會有一些地方需要注意。

1. 關於標點符號
   在數學模式中，我們要注意一下標點符號的問題，一般而言，數學式後面如果有標點符號，在隨文數式，這個標點符號不能納入數學模式中；反之，在展式數式的場合，這些標點符號則要納入數學模式中。例如：

```
Let $f(x)=\sqrt[4]{x+1}$ and $g(x)=\sqrt{9-x^2}$,... % 逗點不納入數學模式
Let
\[
f(x)=\sqrt[4]{x+1}
\]
and
\[
```

```
g(x)=\sqrt{9-x^2},          % 逗點納入數學模式，標點符號也獨立成行
\]
...
```

所以，展式數式，如果數學式最後有個句點或逗點的話，請不要懷疑，你搞對了！:-)

2. 數學模式的斜體字

   數學模式裡頭，預設會使用斜體字，但這些斜體字是數學斜體，和一般文章中的斜體是不一樣的，他字母間的距離比較寬，也沒有所謂的連體字（ligature），因此，如果需要這些效果，可以指定要使用斜體字，這樣就會表現和一般文章一樣的斜體了。例如：

   ```
   \textit{proffer} normal italic.  % 正常文章的斜體
   $proffer\ math\ mode\ iatlic.$   % 數學斜體
   $\textit{proffer\ math\ mode\ normal\ italic.}$ % 指定爲正常斜體
   ```

   表現出來會是：

   proffer normal italic.
   $prof\,fer$ *math mode iatlic.*
   proffer math mode normal italic.

   當然，這種情形很少發生，正常排版的話，無需特別去指定使用一般文章的斜體。

3. 例外不使用斜體字的情形

   一般函數名是不使用斜體字的，例如 log、三角函數名 …… 等等，爲了避免失誤打錯，可以直接使用指令的方式，例如 \log、\sin、\tan……等等，這樣雖然是在數學模式中，也會使用一般的正常字體。TEX/LATEX 系統提供了預先定義好的 32 種函數名供使用：

   ```
   \arccos  \cos   \csc  \exp   \ker     \limsup  \min
   \arcsin  \cosh  \deg  \gcd   \lg      \ln      \Pr
   \arctan  \cot   \det  \hom   \lim     \log     \sec
   \arg     \coth  \dim  \inf   \liminf  \max     \sin
   \sinh    \sup   \tan  \tanh
   ```

   這樣往後只要是函數名就直接在數學模式中使用這些現成的指令就行了。當然，如果是這裡沒有涵蓋的函數名，就得自行加以注意了。

   另外，單位名、化學元素、數字、簡寫縮寫文字等都不使用斜體字。但例外的例外，物理中的常數名則仍然是要使用斜體字，例如光速 c。

4. 不要加入換行指令或插入空白行
   一個數學式子，在 LaTeX 是視爲一個單獨的單位或段落，而且，在這個特殊的段落裡，LaTeX 會抑制 line break 及 page break 的機制，所以，除非是矩陣及矩陣方程式外，不能去強迫換行也不能插入空白行。

10.2 數學符號

我們打字，通常是無法打出一些特殊數學符號，縱使字型裡頭有這種符號，但由於要和其他符號、文字調整他們的相對位置，因此，除了一些常用的運算符號外，數學符號通常是使用指令的方式來鍵入。我們在第 7.2.8 小節裡頭曾提到 `symbols-a4.pdf` 符號表，這個總表非常重要，幾乎羅列了目前所有可用的現成符號，並且會標明需要引入什麼 package，所以，這裡就不把符號表列出來，以節省篇幅。

當然，有些編輯器的巨集會設定好方便的按鈕方式來插入這些數學符號，但建議開始接觸的時候，多花點時間親自鍵入，等熟悉以後再來使用這種方便的設定來增加生產力。理由是，編譯錯誤或校稿修改時才知道要改什麼地方，連 `\sum`、`\infty`、`\int` 是什麼符號都不知道的話，那麼要微調就變成很困難了，而且什麼地方錯誤也常會搞不清楚，這些後續的動作所花的時間，可能會比你剛開始學指令所花的時間還多。

10.3 各種數學式子的書寫方法

我們這裡就正式來看看數學式子到底是如何書寫，這裡不做符號的列表，直接舉例子，如對相關符號指令的書寫有疑問，請自行查閱 `symbols-a4.pdf`。

10.3.1 分式（fraction）

書寫方式	排版結果
`$f(x,y)=3(x+y)y/(2xy-7)$`	$f(x,y)=3(x+y)y/(2xy-7)$
`$f(x,y)=\frac{3(x+y)y}{(2xy-7)}$`	$f(x,y)=\frac{3(x+y)y}{(2xy-7)}$

簡單的分式，直接使用 `/` 就可以了，否則就要使用 `\frac{分子}{分母}` 這個分式的的指令。我們可以看到，隨文數式爲了和前後文配合，分式的情形會把字母、符號縮小，如果太複雜的分式，就不適合使用隨文數式了，要把他單獨列出來：

```
\[
f(x,y)=\frac{3(x+y)y}{(2xy-7)}
\]
\begin{equation}  % 有編號的情形
f(x,y)=\frac{3(x+y)y}{(2xy-7)}
\end{equation}
```

這樣，排版出來的結果是：

$$f(x,y)=\frac{3(x+y)y}{(2xy-7)}$$

$$f(x,y)=\frac{3(x+y)y}{(2xy-7)} \tag{10.1}$$

請注意，和上下行的原來文字不要有空白行，展式數式會自動處理。更複雜的分式的例子：

```
\[
\frac{\frac{a}{x-y}+\frac{b}{x+y}}{\frac{x-y}{x+y}+\frac{a-b}{a+b}}
\]
```

排版出來的結果是：

$$\frac{\frac{a}{x-y}+\frac{b}{x+y}}{\frac{x-y}{x+y}+\frac{a-b}{a+b}}$$

如果覺得，字體似乎太小了，可以指定字體：

```
\[
\frac{\frac{\displaystyle a}{\displaystyle x-y}+
\frac{\displaystyle b}{\displaystyle x+y}}
{\frac{\displaystyle x-y}{\displaystyle x+y}+
\frac{\displaystyle a-b}{\displaystyle a+b}}
\]
```

排版出來的結果是：

$$\frac{\frac{a}{x-y}+\frac{b}{x+y}}{\frac{x-y}{x+y}+\frac{a-b}{a+b}}$$

可指定的字體大小有：

`\displaystyle`	展示數式的標準字體大小
`\textstyle`	隨文數式的標準字體大小
`\scriptstyle`	第一層上下標字體大小
`\scriptscriptstyle`	第二層上下標字體大小

10.3.2 上下標

書寫方式	排版結果
`$(a+b)^2=a^2+2ab+b^2$`	$(a+b)^2=a^2+2ab+b^2$
`$\cos 2x=\cos^2x-\sin^2x$`	$\cos 2x=\cos^2 x-\sin^2 x$
`$(y^m)^n=y^{mn}$`	$(y^m)^n=y^{mn}$
`$^a_bY^c_d$`	${}^a_bY^c_d$
`$e^{t \cos\theta}$`	$e^{t\cos\theta}$
`$\lim_{n \to \infty}\sum_{i=1}^n{\frac{1}{n}}$`	$\lim_{n\to\infty}\sum_{i=1}^n\frac{1}{n}$
`$y_1=1/3(x_1+\omega x_2+\omega^2x_3)$`	$y_1=1/3(x_1+\omega x_2+\omega^2 x_3)$

需要注意的是，不管是上下標，如果裡頭有兩個以上的字元都要當做上下標時，要使用大括號把他括住，否則會只作用在第一個字元而已。而且，上下標是左右兩邊都能標註的。這裡的 \to 指令，是 \rightarrow 的簡寫，就是向右的單箭號。相對的向左的單箭號 \leftarrow，他的簡寫是 \gets。

另外，像 \sum、\lim、\int 這類符號，如果是在展示數式的時候，他的上下標的表現會和隨文數式不一樣，符號也會比較大，例如：

```
\[
\lim_{n \to \infty}\sum_{i=1}^n{\frac{1}{n}}\sin\frac{k}{n}
\]
```

表現出來會變成：

$$\lim_{n\to\infty}\sum_{i=1}^n\frac{1}{n}\sin\frac{k}{n}$$

10.3.3 根號

書寫方式	排版結果
`$\sqrt{x^2+y^2}$`	$\sqrt{x^2+y^2}$
`$\sqrt[5]{a+\sqrt{b}}$`	$\sqrt[5]{a+\sqrt{b}}$

10.4 矩陣（array）

矩陣的排版方式和第 8.3 節所談的 tabular 表格類似，也是以 & 來區隔欄位，以 \\ 來換行，只不過，矩陣的情形是在數學模式裡頭。

其中的分界符號（delimiter），在 LaTeX 是由 \left 及 \right 指令來引導，這些分界符號會隨裡頭式子的多寡，自動調整大小。至於有什麼分界符號可以使用，也請自行查一下 symbols-a4.pdf。我們來看例子：

```
\[
A = \left(   % 視 \left 後面跟的是什麼分界符號，就是使用什麼
\begin{array}{clr}
  t_{11} & t_{12} & t_{13} \\
  t_{21} & t_{22} & t_{23} \\
  t_{31} & t_{32} & t_{33}
\end{array}\right)
\]
```

排版結果是：

$$A = \left(\begin{array}{clr} t_{11} & t_{12} & t_{13} \\ t_{21} & t_{22} & t_{23} \\ t_{31} & t_{32} & t_{33} \end{array} \right)$$

是不是和排版表格一樣呢？指定的位置 lcr 的意思，和排版表格時是一樣的。如果右邊不需分界號，那麼可以使用 \right. 來代替，例如：

```
\[
g(x,y) = \left\{\begin{array}{ll}
                  f(x,y), & \mbox{if $x<y$} \\  % 文字的部份要用 \mbox
                  f(y,x), & \mbox{if $x>y$} \\  % 包住，讓他使用正常字體
                  0,      & \mbox{otherwise.}
                \end{array} \right.
\]
```

排版的結果是：

$$g(x,y) = \left\{ \begin{array}{ll} f(x,y), & \mbox{if } x<y \\ f(y,x), & \mbox{if } x>y \\ 0, & \mbox{otherwise.} \end{array} \right.$$

$\mathcal{AMS}$-LaTeXindexAmS-LaTeX@$\mathcal{AMS}$-LaTeX 另外提供了 matrix 環境，這是沒有分界符號的，另外有固定分界符號的環境：

環境	分界符號	環境	分界符號
matrix	無分界號	bmatrix	方括號 [ ]
pmatrix	小括號 ( )	Bmatrix	大括號 { }
vmatrix	單垂直線 | |	Vmatrix	雙垂直線 || ||

這樣就可以不必使用 \left 及 \right 來引導出分界符號，我們把上面的例子再排版一

次：

```
  ...
\usepackage{amsmath}            % 要記得引用 amsmath 巨集
  ...
\[
A =
\begin{pmatrix}                 % 使用小括號
  t_{11} & t_{12} & t_{13} \\
  t_{21} & t_{22} & t_{23} \\
  t_{31} & t_{32} & t_{33}
\end{pmatrix}
\]
```

排版出來的結果是：

$$A = \begin{pmatrix} t_{11} & t_{12} & t_{13} \\ t_{21} & t_{22} & t_{23} \\ t_{31} & t_{32} & t_{33} \end{pmatrix}$$

看起來會比較緊湊一點。matrix 環境仍然可以使用 \left 及 \right 來加上分界號。

10.4.1 矩陣方程式

LaTeX 有一個內建的 eqnarray 環境，來排版矩陣方程式，但這個環境對數學式子的空間安排會有潛在的瑕疵，因此，許多 LaTeX 專家建議使用 $\mathcal{AMS}$-LaTeX 所提供的 align 環境。因此，這裡只討論 align 環境。而且，使用 align 環境的好處是，每一個數學式子只需要一個 & 即可，排版出來會依這個 & 來對齊，如果有數個數學式在同一行，那各個數學式子也是使用 & 來區隔。所有式子要向左對齊的話，只要把 & 置於行首就可以了。

align 環境中的每行數學式子都會加以編號，要不編號的話，可在每行數學式子換行符號前加個 \notag 指令，這一行便不會編號，要所有的數學式子都不編號的話，就使用 align* 環境。要使用特別指定的符號來編號的話，可使用 \tag{符號} 放在這一行的換行指令前。使用 subequations 環境則會有子編號。我們來看個例子：

```
  ...
\usepackage{amsmath}        % 要記得引用 amsmath package
  ...
\begin{subequations}        % 讓編號同數，但以英文小寫爲子編號
\begin{align}               % 韓信點兵，同餘方程式
x & \equiv 2 \pmod 3 \\     % 三三數之剩二
x & \equiv 3 \pmod 5 \\     % 五五數之剩三
x & \equiv 2 \pmod 7        % 七七數之剩二
```

```
\end{align}
\end{subequations}
```

排版出來的結果是：

$$x \equiv 2 \pmod 3 \tag{10.2a}$$
$$x \equiv 3 \pmod 5 \tag{10.2b}$$
$$x \equiv 2 \pmod 7 \tag{10.2c}$$

10.5 定理

LaTeX 有一個 \newtheorem 指令來定義，Theorem, Lemma, Definition⋯⋯ 等等環境，但他的功能算是滿陽春的，因此，這裡也附帶說明美國數學協會的 amsthm package 的使用，他仍然是建立在 LaTeX 的 \newtheorem 而來的，因此可以配合 LaTeX 原有的定義。

10.5.1 原始 LaTeX 的定義

定理環境，在 LaTeX 裡頭是要我們自行去指定環境名稱，他的語法是：

```
\newtheorem{環境名稱}{定理名稱}[章節層次]
```

編號如要加入章節編號，可加入選項參數的部份，例如 chapter 則會按 chapter 的編號來成爲定理、定義的編號。我們來看看實際的例子：

```
 ...
\newtheorem{defi}{Definition}  % 在 preamble 區先定義好環境名稱
 ...
\begin{defi}
Let $f$ be continuous on the half-open interval $[a, b)$ and suppose
$\lim_{x\rightarrow b^-}|f(x)|=\infty$. Then,
\[
\int_a^bf(x)dx=\lim_{t\rightarrow b^-}\int_a^tf(x)dx
\]
provided this limit exists and is finite, in which case we say the
integral converges. Otherwise, we say it diverges.
\end{defi}
```

表現出來的將會是：

Definition 1. *Let f be continuous on the half-open interval $[a, b)$ and suppose $\lim_{x\to b^-} |f(x)| = \infty$. Then,*

$$\int_a^b f(x)dx = \lim_{t\to b^-} \int_a^t f(x)dx$$

provided this limit exists and is finite, in which case we say the integral converges. Otherwise, we say it diverges.

如果是定理的話，接著可能需要排版證明，我們可以直接使用 amsthm package 的 `proof` 環境，來排版證明的部份。

10.5.2 amsthm 巨集套件

我們在上一個小節裡頭看到了 LaTeX 預設的定理、定義環境，但彈性很小，這裡我們來使用 amsthm 巨集套件，並且試著排版中文：

```
 ...
\usepackage{CJK}                      % 引入 CJK 環境
\usepackage{amsmath,amsthm,amssymb} % 引入 AMS 數學環境
\theoremstyle{remark}                 % 內文使用正常字體
\newtheorem{cdefi}{\bf 定義}          % 改用粗體，預設 remark style 是斜體
 ...
\begin{CJK}{Bg5}{hwmm}
\begin{cdefi}
設函數 $f:[a,b]\rightarrow\mathbb{R}$ 爲可微分，且 $(f')^2$ 爲可積，則稱
\[
L(s)=\int^b_a\sqrt{1+(f'(x))^2}dx
\]
爲 $f$ 之圖形自點 $(a,f(a))$ 至點 $(b,f(b))$ 之弧 $s$ 的弧長。
\end{cdefi}
\end{CJK}
```

排版出來的結果是：

定義 1. 設函數 $f : [a, b] \to \mathbb{R}$ 爲可微分，且 $(f')^2$ 爲可積，則稱

$$L(s) = \int_a^b \sqrt{1 + (f'(x))^2}dx$$

爲 f 之圖形自點 $(a, f(a))$ 至點 $(b, f(b))$ 之弧 s 的弧長。

其中 `theoremstyle` 有三種 style，`plain`, `difinition`, `remark`，預設使用的是 `plain`，即定理名稱會用粗體字，內文則是 italic 斜體字，但這在中文並不適當，所以改由 `remark` style 來排版。要注意的是定理環境內的 `plain` style 雖然使用 italic 字型，但他並沒有進

入數學模式，因此裡頭的數學式子仍然要進入數學模式來排版。另外，實數的那個特殊符號 $\mathbb{R}$ 要引用 amssymb package 才會有，這也是美國數學協會發展的。

amsthm 另提供了 `\theoremstyle` 給使用自行定義除了 `plain`、`definition` 及 `remark` 三種形式外的 style。另外，LaTeX 本身的工具組也提供了一個 theorem 巨集套件，能更有彈性的來細部微調，也有現成的指令，可以去改變所要使用的字體，可以參考系統上就有的 `theorem.dvi` 這個說明檔。

10.6 數學模式中的字型及空間調整

我們前面已有談到調整數學模式字體大小的四個指令。這裡我們再來看看其他的調整指令。

10.6.1 數學字體的改變

以下的指令，相信大家從他的簡寫就可以知道意思：

指令	作用	實例
`\mathrm`	正常字體	ABCabc
`\mathtt`	打字機字族	`ABCabc`
`\mathbf`	粗體字	**ABCabc**
`\mathsf`	sans serif	ABCabc
`\mathit`	italic 斜體	*ABCabc*
`\mathcal`	數學花體字	$\mathcal{ABC}$

其他的套件會有更多的不同字體，請參考系統裡頭的 `symbols-a4.pdf`。要注意的是，有些字型並不是都完全有各種字體組合，像花體字並沒有小寫字母。

10.6.2 數學模式中調整間距

正常情況下，數學模式中的空間調整應不必使用者去操心，但程式畢竟不會思考，有些特殊場合仍然需要人爲的調整。

指令	作用	指令	作用
`\quad`	空出一個 em 單位的空白	`\qquad`	空出兩個 em 的空白
`\,`	加入 1/6 quad 的空白	`\!`	減去 1/6 quad 的空白
`\;`	加入 5/18 quad 的空白	`\:`	加入 2/9 quad 的空白

其中 \,、\quad 及 \qquad 可以用在一般的文字模式及數學模式，其他的只能使用在數學模式中。我們把上一個弧長定義的例子來試著調整一下：

```
 ...
\begin{cdefi}
設函數 $f:[a,b]\rightarrow{\mathbb R}$ 爲可微分，且 $(f')^2$ 爲可積，則稱
\[
L(s)=\int^b_a\!\sqrt{1+(f'(x))^2}\,dx
\]
爲 $f$ 之圖形自點 $(a,f(a))$ 至點 $(b,f(b))$ 之弧 $s$ 的弧長。
\end{cdefi}
 ...
```

排版出來的結果是：

定義 1. 設函數 $f:[a,b]\rightarrow\mathbb{R}$ 爲可微分，且 $(f')^2$ 爲可積，則稱

$$L(s)=\int^b_a\!\sqrt{1+(f'(x))^2}\,dx$$

爲 f 之圖形自點 $(a,f(a))$ 至點 $(b,f(b))$ 之弧 s 的弧長。

但是，要注意的是，如果文章有點長，數學式子也不少，那一定要注意整體的一致性，要調整的話就全文相同的地方都要去調整，否則就使用預設值就可以了，至少他不會太離譜。

第 11 章

一篇文章、一本書的完整結構

好了，寫文章最後也要整理成冊，這也是排版系統要負責的部份。如果只是簡單幾百、幾千字的小文章，那很容易，只要個文章題目，章節標題，那也就夠了。但如果是較正式的論文，那可能還有目錄、參考文獻、索引 …… 等等，甚至一本書籍的話，也要有個封面，及送印刷廠時要用到的裁切記號（crop marks）。如果要置放在網頁上的，那還得注意網路超連結互動的問題，所以，這些細節算是滿瑣碎的，但卻是必要的。

當然，個人也並不是什麼排版、印刷的專家，只能談談我所知道的事項，如果需要補充或修正，請有這方面經驗的朋友，不吝提供心得及指正。個人出版，這實際上不是夢，尤其網路發達的今日。

11.1 目錄（Contents）

目錄的問題，如果不講究的話，使用 LaTeX 預設的就行了。就像第 4.4 節所舉的例子一樣。但如果要做調整的話，除非熟悉 LaTeX 巨集的寫法、定義，否則就得使用現成的巨集套件，例如 minitoc 可讓目錄更緊湊，titletoc 更可做相當幅度的調整及美化。

在 LaTeX 文稿內，`\tablofcontents` 可以排版一般的章節目錄。`\listoffigures` 指令可以排版圖目錄，`\listoftables` 指令則可排版表目錄。但圖表的話是指有進入浮動環境，使用 `\caption` 指令，有編號的圖表而言。請注意，這些目錄指令的置放位置會影響實際目錄出現的順序，沒有特殊需求的話，一般的順序是文、圖、表。

11.1.1 更改目錄標題名稱

預設的情形下，在目錄開頭都會有個標題來引導，例如：**Contents**、**List of Figueres** 及 **List of Tables** 等，但是這在中文的情形看起來會不相稱，我們可以去更改預設值。更

改 LaTeX 預設值得視原來這個值是以什麼形式出現，在目錄是以指令定義的形式出現，所以我們要使用 \renewcommand 這個指令去重定義他。

原來的這些 **Contents** 標題是怎麼「弄」出來的呢？如果手頭上沒有相當的參考書籍，可以參考他的原始定義，例如這篇文章是使用 report class，那麼找一下：

```
/usr/share/texmf/tex/latex/base/report.cls  % Unix-like 系統
C:\texmf\tex\latex\base\report.cls           % DOS/Windows 系統
```

這個檔（依安裝的地方不同，可能會有不同的路徑），搜尋 Contents 這個關鍵字，就可以發現，他們原來的定義是：

```
\newcommand\contentsname{Contents}
\newcommand\listfigurename{List of Figures}
\newcommand\listtablename{List of Tables}
```

這樣就清楚了，我們要重指定的是 contentsname、listfigurename 及 listtablename。其他的情形請依此類推。現在我們來把他改成中文：

```
\renewcommand\contentsname{目~錄~}
\renewcommand\listfigurename{圖~目~錄}
\renewcommand\listtablename{表~目~錄}
```

這裡以 CJK 巨集爲例，由於我們需要中文環境，所以這些更改要放在 CJK 環境中，如果只是更改成其他英文字樣，那我們置於 preamble 區就可以了。

11.1.2 目錄的深度

通常，有編號的章節或有 caption 的圖表才會編入目錄中，但如果想讓目錄的結構更細，那麼我們就得更改列入目錄的深度。目錄深度的表現形式是一種計數器（counter），他的名稱是 tocdepth。以這篇文章的 report class 爲例，他的預設值是（請自行查一下 report.cls）：

```
\setcounter{tocdepth}{2}
```

所以會計算到 subsection，以下的就不列入了（請參考第 3.4.4 小節的章節深度標號）。我們只要在 preamble 區，使用 \setcounter 指令去重新指定，就會改變他的目錄深度。

11.1.3 額外的目錄

這是指沒有編入目錄，但想自行加進去的情形，例如：章節指令使用了星號就不會編號，圖表目錄沒有使用 \caption 指令，也不編入目錄了，這時我們可以使用 \addcontentsline 指令來把他們手動加進去。我們來看看文圖表的三種不同情況：

```
\addcontentsline{toc}{章節名}{標題}
\addcontentsline{lof}{figure}{標題}
\addcontentsline{lot}{table}{標題}
```

這樣就會把這些納入目錄，但是，這還是沒有編號的。目錄中所顯示的頁數，就是這些指令（圖表）所在的頁數。

11.2 交互參照（Cross References）

所謂的參照，指的是在文章某處提及某個其他的章節，或某個頁數，甚至是某個圖表，某個數學式子及某個列舉項目，排版系統必需要有這樣的功能來自動達成這種效果，而 LaTeX 本身提供了三個簡單易用的指令來自動處理，他會自動計算相對的章節、頁數。

當然，由於網路的發達，超連結上的交互參照也變得是不可或缺，但 TeX/LaTeX 畢竟是平面排版系統，並沒有這樣的原始功能，但我們可以經由巨集套件來達成這樣的目的，hyperef 巨集套件就是爲此而寫的，這樣就可以讓 LaTeX 排版的結果去轉換成 PDF/HTML 格式的時候，也有超連結的功能。

廣義的來說，包括目錄的參照、文獻參照、註解的參照及外部檔案的參照（例如，參照某個外部檔案的某個章節）都是屬於交互參照的一部份，但這些議題我們另外單獨討論，因爲他不在 LaTeX 所提供的三個基本參照的指令範圍內。

11.2.1 一般的交互參照

LaTeX 提供了三組基本參照的指令：

```
\label{名稱}     % 置放於要被引用之處，以一個名稱來標記他
\ref{名稱}       % 引用 \label 所標記處的章節
\pageref{名稱}   % 引用 \label 所標記處的頁數
```

這裡頭的名稱都是自行取名的，但爲了避免重複，個人使用上一般使用上會加入章節或圖

表的代號，例如：

```
 ...
\section{\LaTeX\ 的文稿結構}
\label{sec:struct}
 ...
\begin{figure}
\includesgrapics{fontstruct}
\caption{字型的結構}
\label{fig:struct}
\end{figure}
 ...
請參考第 \ref{sec:struct} 節，頁 \pageref{sec:struct}。
請參考圖 \ref{fig:struct}，頁 \pageref{fig:struct}。
 ...
```

這兩個 `struct` 代表不同的參照處，當然，盡量避免這種情形發生，可加入 `fontstruct` 之類的來區別，但前面冠上章節、圖表的簡名，有助於看文稿時清楚區別。請注意，其中 `sec:`、`fig:` 都不是必要的，只是這樣比較容易辨識，而且不容易名稱重複。

要非常注意的的幾個重點是：

1. 有參照的文稿一定要編譯兩次才能正常顯示。
2. 能編號的章節、圖表、列舉項目、數學式、定理才能參照，雖然他們不一定要編號。
3. 圖表的參照 `\label` 一定要在 `\caption` 之後，不能在前。

11.2.2 超連結交互參照（hyperlink）

這當然是要像 PDF/HTML 格式的檔案才有超連結交互參照的可能，像 `*.dvi`、`*.ps` 這類格式的檔案，先天上並沒有這種設計。而 LaTeX 本身並沒有內建這種功能，我們可以使用 hyperref 巨集套件來達成這個目的。現在一般的 TeX 發行版本應該都會附上這個巨集，如果沒有話，可以在以下網站下載、安裝：

ftp://ftp.tug.org/pub/tex/hyperref/

這個套件會讓原本 LaTeX 有交互參照的地方，在製作成 PDF 格式時也會有超連結的功能。他的設定檔是 `hyperref.cfg`，爲了各種文稿使用上的彈性，可以把這個檔在工作目錄上建立一個，這樣會依這個設定檔來執行，可參考本文文稿的原始碼，裡頭會有一個設定檔供參考。當然也是可以在 preamble 區來設定，但這就只能使用在特殊的文稿上了。

他的使用方法，這裡不多做說明，可以參考本文的原始碼裡頭的使用方法，或參考〈由 TeX/LaTeX 製作中文 PDF 檔〉一文：

http://www.study-area.org/tips/latex/chpdf.html
http://www.study-area.org/tips/latex/chpdf.pdf

及 hyperref 所附的使用手冊。

11.3 索引（index）

索引的排版方法上並不算困難，困難的是要選出哪個字詞需要索引，及把各個字詞加入索引指令。我們引用 LaTeX 的標準巨集 makeidx，並在其他加上一個 makeindex 指令，然後在文稿結束前印出索引，下 printindex 就可以了。我們在需要編入索引的名詞後加上 \index{名詞} 經過編譯後就會自動把索引及其相對的頁數計算出來。

11.3.1 索引的結構及編譯

我們來看看文稿裡頭要加入什麼要件：

```
 ...
\usepackage{makeidx}
\makeindex
 ...
要索引的名詞\index{要索引的名詞}
 ...
\printindex         % 一定要有這個指令才會印出索引
\addcontentsline{toc}{chapter}{索引}  % 把他加入目錄
 ...
```

編譯的的程序如下：

```
latex your.tex
makeindex your.idx
latex your.tex
```

11.3.2 索引值的製作

索引值（key）裡頭 |、@ 及 ! 有特殊的意義，要索引他們時前面要加" 來 escape 他。我們來看這些符號實際上有什麼作用：

abc\index{abc}	這是一般正常的索引
xyz\index{abc!xyz}	表示 xyz 是 abc 下的一個子索引
abc\index{abc|textit}	表示這個索引值的頁數使用 italic 斜體排版
abc\index{abc@\textbf{abc}	表示索引值是 abc，但使用粗體排版
\'abc\index{abc@\'abc}	表示依 abc 來排序索引，而不是後面的 ábc

製作索引是需要細心與耐心的，這方面更詳細的資料可以參考系統上的 makeindex.dvi 及 manpages.dvi。要注意的是 \index{} 最理想是緊接在要索引的名詞後，前後都不留空白，有多個 \index{} 相連時亦同，這會讓文件維護增加困難，因此，視每個人的習慣，可以考慮索引在整篇文稿最後才加進去。這份文件也製作了簡單的索引，但這只是當個例子供參考，在製作上有點粗糙，因此，實際上可能會漏掉很多，而且，中文的處理仍有待加強。

製作索引的時候，他的表示法要細心的注意一下，前後相同索引值的表示法要一樣，例如 \index{abc@\textbf{abc} 和 \index{abc@{\bf abc} 這會造成兩個不同的索引，雖然印出來的是一樣。而且，\verb|abc| 這種方式就行不通，因爲 | 這個符號在索引指令內有他的特殊作用，要改用其他的符號代替。如果是和 hyperref 配合的話，abc\index{abc|textit} 也會行不通，因爲 hyperref 對超連結的索引是自動加上 |hyperpage，如果已經有 |textit 了的話，就不會加上去了，這樣一來超連結的部份會被忽略，解決的方法只能去重定義索引方法，或在編譯出來的 *.idx 或 *.ind 上做另外處理。例如：

```
abc\index{abc|textit}
latex 編譯後的情形是：
\indexentry{abc|textit}{143}          % *.idx 檔
經 makeindex 編譯後的情形是：
\item abc, \textit{143}               % *.ind 檔
這樣只編號改變字體，並沒有超連結。而我們要的是：
\item abc, \textit{\hyperpage{143}}   % 這樣才能又變更編號字體又能超連結
```

這個議題比較深入一點，解決的方式可能需要大家一同來研究、研究，不是不能解決，而是方式在使用上是否方便的問題。

索引的處理，他的資訊實際上是產生在編譯後的 *.idx 檔裡頭，然後經由 makeindex 外部程式編譯後，轉換成 *.ind，然後 latex 再次編譯的時候，才把這個 *.ind 引進來，這個 *.ind 其實就是一個 LaTeX 的文稿，他把所有的索引值及頁數，包在一個 theindex 環境中來引入排版的。

說明這些的用意就是暗示，我們可以由外部處理程式去動手腳，把索引的部份再「加工」，包括中文資料的處理也是一樣，下一節要談到的參考文獻的處理機制也是類似的情形，

而這也就是爲什麼 latex 要執行好幾次的原因，也正因爲這樣，我們才有動手腳的機會，例如 makeindex 就有 -s 參數，可以接受外部的 style 檔，或者，如果工作目錄上有 *.mst（makeindex style）這個檔，也會優先去參考錢，這樣就可以產生不同形式的索引結果。當然，參考文獻可以另單獨的一個文獻外部檔，索引的話，目前則沒有辦法這樣做，是否可以比照參考文獻的做法，由外部檔案來處理呢？就請大家腦筋急轉彎一下了，這樣也可以讓文稿更容易維護。

11.3.3 更改索引標題

預設的索引標題是 **Index**，我們可以在 preamble 區來更改他（中文的話，請放在本文區 CJK 環境內），例如，設要中文名稱的話，可更改爲中文：

```
\renewcommand\indexname{索~引}
```

11.4 參考文獻（Bibliography）

參考文獻可以經由 LaTeX 內建的 thebibliography 環境來製作。長篇文稿，也可以使用 BibTeX 由外部檔案來製作。至於參考文獻的格式，就要符合邀稿單位的規格了，這裡不多做說明。

11.4.1 thebibliography 環境

在進入 thebibliography，編譯後他會自成一個獨立的章節，如果是 article 類別的文稿，他會自動印出 **Referrences** 的字樣爲標題，如果是 report 或 book 類別的文稿，他會印出 **Bibliography** 的字樣爲標題。

在 thebibliography 環境裡頭，他是由 \bibitem 指令來列出資料的，我們來看一下他的語法：

```
\begin{thebibliography}{99}  % 參考文獻印出之編號最寬爲兩個字母寬
\bibitem[標記一]{鍵值一} 參考資料一
\bibitem[標記二]{鍵值二} 參考資料二
 ...
\edn{thebibliography}
```

所謂的「標記」這是選項參數，如果沒有的話，則正常引用後會在甲用處使用阿拉伯數字

外加方括號來顯示；如果有加入的話，引用後會使用所加入的標記來顯示。那個「鍵值」指的就是引用時的關鍵字，後面所接的「參考資料」就是書籍、論文等資訊。其中的 {99} 只表示在最後的參考文獻印出來的時候，最開始的編號統一在兩個字母寬，如果都沒有使用「標記」，那麼就是兩個數目字寬，如果有使用「標記」，那麼要設在最長標記的字母寬，否則印出時會無法對齊。

我們引用的時候是使用 \cite{鍵值} 這個指令，他會顯示參考資料中的編號值，且以方括號括起來。我們來看看一些設定及引用的例子，:

```
\begin{thebibliography}{KDE}  % 參考文獻中印出的編號最寬爲三個字母寬
\bibitem{KDEt} Knuth, D.E., \textit{The \TeX{}book},
Reading, Massachusetts: Addison-Wesley, 1989.
\bibitem[KDE]{KDEm} Knuth, D.E., \textit{The \MF{}book},
Reading, Massachusetts: Addison-Wesley, 1986.
 ...
\end{thebibliography}
```

參考文獻印出的結果請參考本文件後面關於參考資料的部份。至於引用方式及其引用情形如下（顏色的部份是因爲使用 hyperref 套件的超連結）：

引用方式	引用結果
請參考 \cite{KDEt}	請參考 [1]
請參考 \cite[1989]{KDEt}	請參考 [1, 1989]
請參考 \cite{KDEm}	請參考 [KDE]
請參考 \cite[1986]{KDEm}	請參考 [KDE, 1986]
請參考 \cite{KDEt,KDEm}	請參考 [1, KDE]

如果你現在是在觀看 PDF 格式的檔案，那在 Xpdf 或 Adobe Acrobat Reader 都可以使用滑鼠來點一下，看看眞正在後面印出時是什麼樣子。再次強調，參考文獻的規格請依邀稿單位的要求，這裡所列出來的不是「標準」。

11.4.2 更改標題名稱

前面已提到過更改目錄及索引的標題，同樣的方法，我們也可以更改參考資料的標題，只是要注意引用的文稿類別是什麼。

```
\renewcommand\refname{參~考~資~料}  % article 類別文稿
\renewcommand\bibname{參~考~文~獻}  % report/book 類別文稿
```

11.4.3 BibTEX 簡介

如果常常有寫論文的機會，整理出自己的一份參考文獻資料庫可以節省許多時間，正常情況下，使用 bibtex 來處理外部文獻檔案的情形，只有引用到的文獻才會印出來，這樣也就不必擔心印出一堆不相關的文獻了。另外一個好處是，這個參考文獻資料庫可以另外獨立維護，所有的文章都用這一份資料庫，這在維護上會很方便，也減少錯誤的機會。

BibTEX 本身提供一個外部的 bibtex 工具程式，在 latex 編譯過文稿後，再利用 bibtex 編譯一次文稿，最後再使用 latex 重編譯過。而參考文獻資料庫是按一定的格式寫於 *.bib 檔案裡頭，在文稿中則以 \bibliogrphy 指令來引入，編譯過程中自然會去參考這個外部考文獻資料庫。他的使用情形如下（以 your.bib 爲例）：

```
 ...
\begin{document}
\bibliogrphiystyle{plain}  % 指定 style 檔
 ...
\bibliography{your.bib}    % *.bib 延伸檔名可以省略
 ...
\end{document}
編譯過程：
latex example
bibtex example
latex example
```

11.4.3.1 *.bib 檔的格式

*.bib 檔的格式自成一格，和寫在原來文稿裡頭的不同，視資料的性質，要把他標明出來，例如書籍類是 @book 來開頭，期刊文章使用 @article 來起頭，我們來看一個例子：

```
@book{ KDEt,
author      = "Knuth, Donald E.",
year        = "1989",
title       = "The {\TeX}book",
publisher   = "Addison-Wesley",
address     = "Reading, Massachusetts",
volumn      = " ",
edition     = " ",
month       = " ",
series      = " ",
note        = " ",
}
@article{ somekey,
author      = "Someone",
```

```
year         = "2004",
title        = "The {\TeX} Journal",
journal      = "SayYa-Publisher",
volumn       = " ",
number       = " ",
pages        = " ",
month        = " ",
note         = " ",
}
```

每行後的逗點是必要的，名字的話 Knuth, Donald E. 或 Donald E. Knuth 這兩種方式，bibtex 都能認得，但姓擺在前面的時候其後要加個逗點，如果是兩位以上的作者時要以 and 來連接。雖然可以使用 LaTeX 的語法，這時他整個要由大括號括起來，而且，註解符號 % 不被接受。紅色的項目是必要的項目，其他項目可以列進去，也可以省略，要加進去的話，則以 " " 空出來，這樣以後有這方面的資料時再填進去。

顯現的形式是受 *.bst 格式檔在控制的，所以，不必要的標點符號不要自行加進去，書名的字體顯示也無需加進去。

引用的方式同樣是使用 \cite 指令，一般只要要引用到的資料才會印出來，如果要全部 *.bib 裡頭的資料都印出來的話，可以加個 \nocite{*} 指令。

在使用中文的情形下，bibtex 程式認不得中文，在 CJK 環境下編譯會出問題，我們可以先編輯一個 *.cbib 檔，然後再使用 bg5conv 來把他轉成 *.bib 檔：

```
bg5conv < some.cbib > some.bib
```

這樣，在文稿裡頭引入 some.bib 就可以了。

11.4.3.2 格式檔

BibTeX 的格式檔是 *.bst（bibliography style），我們上面所引用的是 plain 其實就是引用 plain.bst 這個格式檔，這是最基本的格式，在編譯時期會依這個格式檔來印出參考文獻的顯現形式。其他尚有：

plain	依字母的順序印出，比較順序爲 author, year, title
unsrt	依引用的先後次序印出
abbrv	與 plain 相同，但 first name, month, title, journal 以縮寫印出
alpha	引用處顯示 [作者年份] 來取代數目字。

已經有許多人發表過特定的格式檔，但這些對於中文則無法完全合乎我們的使用習慣，

例如標點符號及書名號，但我們可以去更改他們的格式，這方面的資料請參考系統上的 btxdoc.dvi 及 btxhak.dvi 這兩個說明檔。

這也算是目前的一個值得去研究的空間，尤其是中文及 Unciode 編碼文件索引、排序及排版的問題，這在英文語系算是比較容易解決，都有現成的格式範例可以運用，但中文就比較缺乏這方面的範例。在吳聰敏教授的《cwTEX 排版系統》[5] 一書裡頭，有對這方面做過努力，使用的是外部程式工具 cwbibtex，再和 cw.bst 格式檔配合的話，有不錯的結果。

11.5 附錄（Appendix）

排版附錄使用的是 \appendix 指令，這個指令以後和正常一般編輯即可，不同的附錄以 \chapter 來區隔，但印出來的時候會標上大寫字母，而不是原來的 **Chapter** 字樣。也是可以有 \section 指令，這資了大寫英文字母外，會緊接著附上阿拉伯數字，例如 **A.1**、**B.2** 等等。

11.5.1 改變附錄的標題

在英文環境，附錄是以 **Appendix** 爲標題來開始的，在中文環境下我們要把他改成中文：

```
\renewcommand\appendixname{附~錄}
```

article 類別的文稿並不印出 **Appendix** 字樣，因此也就沒有 appendixname 來更改。

11.6 大型文稿的維護

通常我們寫一篇文章，大概都是一個文稿寫到底，但如果超過一百頁的文稿的時候，維護起來會比較困難，所以 LATEX 提供了 \input 及 \include 指令來將外部檔案引進來，當做是文稿的一部份來編譯，這樣就可以按章節來把文稿分開處理、維護。

他們的使用方法很簡單，一個是含主要正常結構的主檔，其他的則沒有 preamble 區，只有本文區，就是按照一般文稿中的章節的部份來書寫就可以了。以這一篇文件的例子來說，他的主檔是 latex123.tex，主要的內容如下：

```
\documentclass[12pt,a4paper]{report}
 ...
```

```
 這裡是 preamble 區的內容
 ...
\begin{document}
\begin{CJK}{Bg5}{hwmm}
 ...
%begin{latexonly}
\input{story.cjk}
\input{preparation.cjk}
\input{syntax.cjk}
\input{start.cjk}
\input{space.cjk}
\input{class.cjk}
\input{package.cjk}
\input{table.cjk}
\input{graphic.cjk}
\input{math.cjk}
\input{abook.cjk}
\input{theend.cjk}
\input{fdl.cjk}
%end{latexonly}
\begin{htmlonly}
\input{story}
\input{preparation}
\input{syntax}
\input{start}
\input{space}
\input{class}
\input{package}
\input{table}
\input{graphic}
\input{math}
\input{abook}
\input{theend}
\input{fdl}
\end{htmlonly}
 ...
\end{CJK}
\end{document}
```

這裡分成兩段來處理是因爲要使用 bg5latex 來編譯文稿，但 latex2html 本身已經可以處理未經前置處理的 LaTeX 中文文稿，所以分成兩個部份來分別處理。\input{} 裡頭的案名，如果沒有加附檔名的話，預設他的延伸檔名是 *.tex。CJK package 另外提供了 \CJKinput 及 \CJKinclude 兩個指令，也可以用來引入中文文稿。

11.6.1 input 和 include 的差異

這兩個指令都可以用來引進外部文稿，但有一些細節不一樣。\input 的情形，他可以另外指定延伸檔名，\include 則不行，他一定要是 *.tex 的延伸檔名。在引入的時候 \include 會起新頁，\input 則不一定，要視文稿類別而定。最重要的差異在於 \include 可以在 preamble 區和 \includeonly{} 指令配合，這樣就不必每次都要編譯整份文稿，只編譯新文稿就可以了，頁數、參照還是會正確顯示，這在排版大型書籍的時候就很好用了。

11.7 裁切記號（crop marks）

裁切記號是用在排版完成，送印後會有個標記符號，讓印刷廠可以跟據這些標記符號來做截切、剪裁的工作。LaTeX 本身並沒有內建這種功能，這得和 crop package 來配合。

由於個人也並沒有這方面的實際經驗，因此這裡不多做說明，請自行參考 crop 巨集套件的說明。

第 12 章

後記

大概簡單介紹了 LaTeX 的使用，這樣夠用了嗎？很可能是不夠的，尤其是想變動風格的時候，但一般使用，不講求花俏，應該可以用了，剩下的只是熟練的問題。當然，許多細節可能並沒有說明清楚，但方向知道了，其他的自行查閱就行了。眞碰到問題時，可以到 bbs/news 上詢問。

有許多沒有提及的東西，在這一章交代一下，有些是可以自行查閱的，有些則是尚未成熟，可能還不到眞正實用的階段（也可能是我自己也不會用啦！:）。

1. 微調
   由於微調牽涉到對 LaTeX macro 的一定程度的認識，因此並沒有說明得很清楚，理想的話應該先把 TeX/LaTeX 巨集的寫法先做簡單的介紹，這個部份可能另外專文介紹較妥，畢竟這篇文件是定位在入門級教材。

2. 中文的處理
   中文的處理還有很多模糊地帶，例如，索引、參考文獻及中文直排。目前的其他中文 TeX/LaTeX 也沒有介紹，例如 ChiTeX、cwTeX 及 PUTeX 等等。

   TeX/LaTeX 系統的字型機制算是較複雜，安裝字型更是一般使用者的夢魘，英文寫作比較容易解決，通常系統上都會安裝好，中文的話就比較麻煩，除了詳細去介紹外，我們使用中文字型應該有個大家認同的規格才行，否則我這份文件拿到其他的中文 TeX/LaTeX 系統上的時候，就必須修改一下，至少換個字型名稱才能順利編譯。

3. 實例嫌不夠完整
   尤其是表格、圖形處理及數理排版的部份，並沒有交待得很完整。這裡頭當然牽涉到許多的背景知識的問題，不單純在排版本身，這在其他的排版系統一樣會碰到同樣的情況。當然，有很大的部份是我個人經驗不足的關係啦！:-)

4. 各種檔案格式的介紹
   TEX/LATEX 系統中的檔案格式多如牛毛，包括一些中間產生的檔案都有他特定的目的，但我們並沒有多做介紹。原因是，這些檔案都牽涉到他的運作機制，這麼一來連運作機制也要說明才行，這樣會使篇幅大增，而且也會擾亂了初學者的學習步調，因此，只能留待往後有機會再來介紹。

5. 現有巨集庫的整理
   LATEX 的巨集實在是太多了，現有資料大多是英文的，是有必要整理出一份有系統的中文速查表，以免重複去製造輪子。

6. 重音符號、歐洲字元
   這些都沒有眞正接觸到。這些內容，個人不敢造次，因爲並不很熟悉，因此，得要有懂歐洲語系的朋友來個完整的介紹才行。

7. Unicode 編碼文件的處理
   這方面也沒介紹，但由於這是一篇入門級的 LATEX 教學文件，這方面的內容應該是由另外的專文來介紹可能會比較恰當。

8. 投影片的製作
   投影片、幻燈片的議題目前很流行，LATEX 也是可以製作精美的投影片，這方面的文件可以參考：

   http://www.miwie.org/presentations/presentations.html

   當然，以上的資料是使用在英文語系，中文的話，我們得另做介紹。

9. 和 XML/SGML/HTML 的配合
   這個完全沒有提到，一方面 XML/SGML 的內容沒有想像中簡單，他們的應用範圍實在是太廣了，這方面的內容得慢慢來補充。

10. 和資料庫系統的配合
    這方面在目前 seaching 掛帥的 Internet 是相當重要的課題，這當然是不適合放在這份文件，得另外在進階的文件做更進一步的介紹。TEX 系統的生命力、可塑性相當強，因此和其他的文件系統的結合能力也就比較容易達成。

11. GUI 圖形界面
    這個部份也沒有深入介紹，最重要是卡在中文的問題上，這方面需要有興趣的朋友共同來研究，雖然命令列環境的生產效率很高，但爲了顧及一般使用者的習慣，方便入門的 GUI 圖形界面的確有其需要。

這份文件，感謝行政院研考會委辦，朝陽大學洪朝貴授與輔仁大學毛慶禎教授共同主持的「政府機關資料文件交換之電子檔案格式應用研究」計畫做部份的補助，很高興他們都能認同自由文件、自由軟體，在目前社會體制下的存在價值。

這份文件的 PDF 格式所嵌入的中文字型，採用的是王漢宗博士所捐贈的三十幾套的 TTF 向量字型，王博士以前就曾捐贈過十三套的 TTF 向量字型，再加上這次的三十幾套，我們自由軟體社群的中文字型就更充實了。更難得的是，這些字型都是採用 GNU GPL 的授權，和這份 GNU FDL 自由文件配合起來，相得益彰，感謝王漢宗博士的慷慨無私及對自由軟體、自由文件的認同。

這份文件採用的是 PDF 及 HTML 格式，這當然得需要網路資源才能呈現在各位面前，感謝 kenduest 及交大數學和 ctshieh 及淡江數學提供這方面的資源，沒有他們的幫忙，文件就無法呈現在各位眼前了！

也希望大家能對這份文件多多的指正及建議。寫作期間，已經接到許多朋友的來信指正，很感謝他們。這份文件的 HTML/PDF 格式及原始文稿，可以在以下網站取得：

http://edt1023.sayya.org/tex/latex123/index.html
http://edt1023.sayya.org/tex/latex123/latex123.pdf
http://edt1023.sayya.org/tex/latex123/latex123-v1.0-src.tar.gz
http://MathNet.math.tku.edu.tw/~edt1023/tex/latex123/index.html
http://MathNet.math.tku.edu.tw/~edt1023/tex/latex123/latex123.pdf
http://MathNet.math.tku.edu.tw/~edt1023/tex/latex123/latex123-v1.0-src.tar.gz

第 13 章

GNU 自由文件許可證原文

GNU Free Documentation License

Version 1.2, November 2002

59 Temple Place, Suite 330, Boston, MA 02111-1307 USA

0. PREAMBLE

The purpose of this License is to make a manual, textbook, or other functional and useful document "free" in the sense of freedom: to assure everyone the effective freedom to copy and redistribute it, with or without modifying it, either commercially or noncommercially. Secondarily, this License preserves for the author and publisher a way to get credit for their work, while not being considered responsible for modifications made by others.

This License is a kind of "copyleft", which means that derivative works of the document must themselves be free in the same sense. It complements the GNU General Public License, which is a copyleft license designed for free software.

We have designed this License in order to use it for manuals for free software, because free software needs free documentation: a free program should come with manuals providing the same freedoms that the software does. But this License is not limited to software manuals; it can be used for any textual work, regardless of subject matter or whether it is published as a printed book. We recommend this License principally for works whose purpose is instruction or reference.

1. APPLICABILITY AND DEFINITIONS

This License applies to any manual or other work, in any medium, that contains a notice placed by the copyright holder saying it can be distributed under the terms of this License. Such a notice grants a world-wide, royalty-free license, unlimited in duration, to use that work under the conditions stated herein. The "Document", below, refers to any such manual or work. Any member of the public is a licensee, and is addressed as "you". You accept the license if you copy, modify or distribute the work in a way requiring permission under copyright law.

A "Modified Version" of the Document means any work containing the Document or a portion of it, either copied verbatim, or with modifications and/or translated into another language.

A "Secondary Section" is a named appendix or a front-matter section of the Document that deals exclusively with the relationship of the publishers or authors of the Document to the Document's overall subject (or to related matters) and contains nothing that could fall directly within that overall subject. (Thus, if the Document is in part a textbook of mathematics, a Secondary Section may not explain any mathematics.) The relationship could be a matter of historical connection with the subject or with related matters, or of legal, commercial, philosophical, ethical or political position regarding them.

The "Invariant Sections" are certain Secondary Sections whose titles are designated, as being those of Invariant Sections, in the notice that says that the Document is released under this License. If a section does not fit the above definition of Secondary then it is not allowed to be designated as Invariant. The Document may contain zero Invariant Sections. If the Document does not identify any Invariant Sections then there are none.

The "Cover Texts" are certain short passages of text that are listed, as Front-Cover Texts or Back-Cover Texts, in the notice that says that the Document is released under this License. A Front-Cover Text may be at most 5 words, and a Back-Cover Text may be at most 25 words.

A "Transparent" copy of the Document means a machine-readable copy, represented in a format whose specification is available to the general public, that is suitable for revising the document straightforwardly with generic text editors or (for images composed of pixels) generic paint programs or (for drawings) some widely available drawing editor, and that is suitable for input to text formatters or for automatic translation to a variety of formats suitable for input to text formatters. A copy made in an otherwise Transparent file format whose markup, or absence of markup, has been arranged to thwart or discourage subsequent modification by readers is not Transparent. An image format is not Transparent if used for any substantial amount of text. A copy that is not "Transparent" is called "Opaque".

Examples of suitable formats for Transparent copies include plain ASCII without markup, Texinfo input format, LaTeX input format, SGML or XML using a publicly available DTD, and standard-conforming simple HTML, PostScript or PDF designed for human modification. Examples of transparent image formats include PNG, XCF and JPG. Opaque

formats include proprietary formats that can be read and edited only by proprietary word processors, SGML or XML for which the DTD and/or processing tools are not generally available, and the machine-generated HTML, PostScript or PDF produced by some word processors for output purposes only.

The “Title Page” means, for a printed book, the title page itself, plus such following pages as are needed to hold, legibly, the material this License requires to appear in the title page. For works in formats which do not have any title page as such, “Title Page” means the text near the most prominent appearance of the work’s title, preceding the beginning of the body of the text.

A section “Entitled XYZ” means a named subunit of the Document whose title either is precisely XYZ or contains XYZ in parentheses following text that translates XYZ in another language. (Here XYZ stands for a specific section name mentioned below, such as “Acknowledgements”, “Dedications”, “Endorsements”, or “History”.) To “Preserve the Title” of such a section when you modify the Document means that it remains a section “Entitled XYZ” according to this definition.

The Document may include Warranty Disclaimers next to the notice which states that this License applies to the Document. These Warranty Disclaimers are considered to be included by reference in this License, but only as regards disclaiming warranties: any other implication that these Warranty Disclaimers may have is void and has no effect on the meaning of this License.

2. VERBATIM COPYING

You may copy and distribute the Document in any medium, either commercially or noncommercially, provided that this License, the copyright notices, and the license notice saying this License applies to the Document are reproduced in all copies, and that you add no other conditions whatsoever to those of this License. You may not use technical measures to obstruct or control the reading or further copying of the copies you make or distribute. However, you may accept compensation in exchange for copies. If you distribute a large enough number of copies you must also follow the conditions in section 3.

You may also lend copies, under the same conditions stated above, and you may publicly display copies.

3. COPYING IN QUANTITY

If you publish printed copies (or copies in media that commonly have printed covers) of the Document, numbering more than 100, and the Document’s license notice requires Cover Texts, you must enclose the copies in covers that carry, clearly and legibly, all these Cover Texts: Front-Cover Texts on the front cover, and Back-Cover Texts on the back cover. Both covers must also clearly and legibly identify you as the publisher of these copies. The front

cover must present the full title with all words of the title equally prominent and visible. You may add other material on the covers in addition. Copying with changes limited to the covers, as long as they preserve the title of the Document and satisfy these conditions, can be treated as verbatim copying in other respects.

If the required texts for either cover are too voluminous to fit legibly, you should put the first ones listed (as many as fit reasonably) on the actual cover, and continue the rest onto adjacent pages.

If you publish or distribute Opaque copies of the Document numbering more than 100, you must either include a machine-readable Transparent copy along with each Opaque copy, or state in or with each Opaque copy a computer-network location from which the general network-using public has access to download using public-standard network protocols a complete Transparent copy of the Document, free of added material. If you use the latter option, you must take reasonably prudent steps, when you begin distribution of Opaque copies in quantity, to ensure that this Transparent copy will remain thus accessible at the stated location until at least one year after the last time you distribute an Opaque copy (directly or through your agents or retailers) of that edition to the public.

It is requested, but not required, that you contact the authors of the Document well before redistributing any large number of copies, to give them a chance to provide you with an updated version of the Document.

4. MODIFICATIONS

You may copy and distribute a Modified Version of the Document under the conditions of sections 2 and 3 above, provided that you release the Modified Version under precisely this License, with the Modified Version filling the role of the Document, thus licensing distribution and modification of the Modified Version to whoever possesses a copy of it. In addition, you must do these things in the Modified Version:

A. Use in the Title Page (and on the covers, if any) a title distinct. from that of the Document, and from those of previous versions (which should, if there were any, be listed in the History section of the Document). You may use the same title as a previous version if the original publisher of that version gives permission.

B. List on the Title Page, as authors, one or more persons or entities responsible for authorship of the modifications in the Modified Version, together with at least five of the principal authors of the Document (all of its principal authors, if it has fewer than five), unless they release you from this requirement.

C. State on the Title page the name of the publisher of the Modified Version, as the publisher.

D. Preserve all the copyright notices of the Document.

E. Add an appropriate copyright notice for your modifications adjacent to the other copyright notices.

F. Include, immediately after the copyright notices, a license notice giving the public permission to use the Modified Version under the terms of this License, in the form shown in the Addendum below.

G. Preserve in that license notice the full lists of Invariant Sections and required Cover Texts given in the Document's license notice.

H. Include an unaltered copy of this License.

I. Preserve the section Entitled "History", Preserve its Title, and add to it an item stating at least the title, year, new authors, and publisher of the Modified Version as given on the Title Page. If there is no section Entitled "History" in the Document, create one stating the title, year, authors, and publisher of the Document as given on its Title Page, then add an item describing the Modified Version as stated in the previous sentence.

J. Preserve the network location, if any, given in the Document for public access to a Transparent copy of the Document, and likewise the network locations given in the Document for previous versions it was based on. These may be placed in the "History" section. You may omit a network location for a work that was published at least four years before the Document itself, or if the original publisher of the version it refers to gives permission.

K. For any section Entitled "Acknowledgements" or "Dedications", Preserve the Title of the section, and preserve in the section all the substance and tone of each of the contributor acknowledgements and/or dedications given therein.

L. Preserve all the Invariant Sections of the Document, unaltered in their text and in their titles. Section numbers or the equivalent are not considered part of the section titles.

M. Delete any section Entitled "Endorsements". Such a section may not be included in the Modified Version.

N. Do not retitle any existing section to be Entitled "Endorsements" or to conflict in title with any Invariant Section.

O. Preserve any Warranty Disclaimers.

If the Modified Version includes new front-matter sections or appendices that qualify as Secondary Sections and contain no material copied from the Document, you may at your option designate some or all of these sections as invariant. To do this, add their titles to the list of Invariant Sections in the Modified Version's license notice. These titles must be distinct from any other section titles.

You may add a section Entitled “Endorsements”, provided it contains nothing but endorsements of your Modified Version by various parties–for example, statements of peer review or that the text has been approved by an organization as the authoritative definition of a standard.

You may add a passage of up to five words as a Front-Cover Text, and a passage of up to 25 words as a Back-Cover Text, to the end of the list of Cover Texts in the Modified Version. Only one passage of Front-Cover Text and one of Back-Cover Text may be added by (or through arrangements made by) any one entity. If the Document already includes a cover text for the same cover, previously added by you or by arrangement made by the same entity you are acting on behalf of, you may not add another; but you may replace the old one, on explicit permission from the previous publisher that added the old one.

The author(s) and publisher(s) of the Document do not by this License give permission to use their names for publicity for or to assert or imply endorsement of any Modified Version.

5. COMBINING DOCUMENTS

You may combine the Document with other documents released under this License, under the terms defined in section 4 above for modified versions, provided that you include in the combination all of the Invariant Sections of all of the original documents, unmodified, and list them all as Invariant Sections of your combined work in its license notice, and that you preserve all their Warranty Disclaimers.

The combined work need only contain one copy of this License, and multiple identical Invariant Sections may be replaced with a single copy. If there are multiple Invariant Sections with the same name but different contents, make the title of each such section unique by adding at the end of it, in parentheses, the name of the original author or publisher of that section if known, or else a unique number. Make the same adjustment to the section titles in the list of Invariant Sections in the license notice of the combined work.

In the combination, you must combine any sections Entitled “History” in the various original documents, forming one section Entitled “History”; likewise combine any sections Entitled “Acknowledgements”, and any sections Entitled “Dedications”. You must delete all sections Entitled “Endorsements”.

6. COLLECTIONS OF DOCUMENTS

You may make a collection consisting of the Document and other documents released under this License, and replace the individual copies of this License in the various documents with a single copy that is included in the collection, provided that you follow the rules of this License for verbatim copying of each of the documents in all other respects.

You may extract a single document from such a collection, and distribute it individually under this License, provided you insert a copy of this License into the extracted document, and follow this License in all other respects regarding verbatim copying of that document.

7. AGGREGATION WITH INDEPENDENT WORKS

A compilation of the Document or its derivatives with other separate and independent documents or works, in or on a volume of a storage or distribution medium, is called an "aggregate" if the copyright resulting from the compilation is not used to limit the legal rights of the compilation's users beyond what the individual works permit. When the Document is included in an aggregate, this License does not apply to the other works in the aggregate which are not themselves derivative works of the Document.

If the Cover Text requirement of section 3 is applicable to these copies of the Document, then if the Document is less than one half of the entire aggregate, the Document's Cover Texts may be placed on covers that bracket the Document within the aggregate, or the electronic equivalent of covers if the Document is in electronic form. Otherwise they must appear on printed covers that bracket the whole aggregate.

8. TRANSLATION

Translation is considered a kind of modification, so you may distribute translations of the Document under the terms of section 4. Replacing Invariant Sections with translations requires special permission from their copyright holders, but you may include translations of some or all Invariant Sections in addition to the original versions of these Invariant Sections. You may include a translation of this License, and all the license notices in the Document, and any Warranty Disclaimers, provided that you also include the original English version of this License and the original versions of those notices and disclaimers. In case of a disagreement between the translation and the original version of this License or a notice or disclaimer, the original version will prevail.

If a section in the Document is Entitled "Acknowledgements", "Dedications", or "History", the requirement (section 4) to Preserve its Title (section 1) will typically require changing the actual title.

9. TERMINATION

You may not copy, modify, sublicense, or distribute the Document except as expressly provided for under this License. Any other attempt to copy, modify, sublicense or distribute the Document is void, and will automatically terminate your rights under this License.

However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

10. FUTURE REVISIONS OF THIS LICENSE

The Free Software Foundation may publish new, revised versions of the GNU Free Documentation License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns. See `http://www.gnu.org/copyleft/`.

Each version of the License is given a distinguishing version number. If the Document specifies that a particular numbered version of this License "or any later version" applies to it, you have the option of following the terms and conditions either of that specified version or of any later version that has been published (not as a draft) by the Free Software Foundation. If the Document does not specify a version number of this License, you may choose any version ever published (not as a draft) by the Free Software Foundation.

ADDENDUM: How to use this License for your documents

To use this License in a document you have written, include a copy of the License in the document and put the following copyright and license notices just after the title page:

> Copyright © YEAR YOUR NAME. Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.2 or any later version published by the Free Software Foundation; with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts. A copy of the license is included in the section entitled "GNU Free Documentation License".

If you have Invariant Sections, Front-Cover Texts and Back-Cover Texts, replace the "with…Texts." line with this:

> with the Invariant Sections being LIST THEIR TITLES, with the Front-Cover Texts being LIST, and with the Back-Cover Texts being LIST.

If you have Invariant Sections without Cover Texts, or some other combination of the three, merge those two alternatives to suit the situation.

If your document contains nontrivial examples of program code, we recommend releasing these examples in parallel under your choice of free software license, such as the GNU General Public License, to permit their use in free software.

參考資料

[1] Knuth, Donald E., *The TEXbook*, Reading, Massachusetts: Addison-Wesley, 1989.

[KDE] Knuth, Donald E., *The METAFONTbook*, Reading, Massachusetts: Addison-Wesley, 1986.

[2] Clark, Malcolm, *TEX by Topic*, Reading, Massachusetts: Addison-Wesley, 1992.

[3] Abrahams, Paul W., Berry, Karl and Hargreaves, Kathryn A., *TEX for the Impatient*, Reading, Massachusetts: Addison-Wesley, 1990.

[4] Goossens, Michel, Mittelbach, Frank, and Samarin, Alexander, *The LATEX Companion*, Reading, Massachusetts: Addison-Wesley, 1993.

[5] 吳聰敏、吳聰慧，《cwTEX 排版系統》，台北：吳聰敏，2002。

[6] 鄭郁耀、郭雅思，《LATEX 數理排版：由入門到應用》，台北：全華，1995。

[7] 王佑中，《Linux 中文應用手冊》，台北：第三波，1995。

[8] 朱浤源，《撰寫博碩士論文實戰手冊》，台北：正中，1999。

[9] 傅祖慧，《科學論文的寫作審查及發表》，台北：中華農學會，1980。

使用許可證聲明

索 引