一份不太簡短的 LaTeX 2ε 介紹

或 *101* 分鐘學會 *LaTeX 2ε*

原版作者： Tobias Oetiker
Hubert Partl, Irene Hyna and Elisabeth Schlegl
原版版本： Version 4.20, May 31, 2006

中文翻譯： 中文 TeX 學會
中文版本： 版本 4.20，二零零七年九月

致謝！

在這份介紹中使用的許多材料來自一個奧地利人使用德語撰寫的 LaTeX 2.09 介紹：

Hubert Partl <partl@mail.boku.ac.at>
Zentraler Informatikdienst der Universität für Bodenkultur Wien

Irene Hyna <Irene.Hyna@bmwf.ac.at>
Bundesministerium für Wissenschaft und Forschung Wien

Elisabeth Schlegl <noemail>
in Graz

如果你對德文文檔有興趣，有一個由 Jörg Knappen 針對 LaTeX 2_ε 更新的版本，在 CTAN 的位置是： CTAN:/tex-archive/info/lshort/german

下列人士爲改進此文提供了校正、建議和素材。他們的不懈努力幫助我把這份文檔實現爲現在這樣子。我對他們所有人表示誠摯的感謝。當然，你在本書中找到的所有錯誤都是我的失誤。而你見到的每一個拼寫正確的單詞，都一定是由於下面列出的這些人之一通知了我。

Rosemary Bailey, Marc Bevand, Friedemann Brauer, Jan Busa, Markus Brühwiler, Pietro Braione, David Carlisle, José Carlos Santos, Neil Carter, Mike Chapman, Pierre Chardaire, Christopher Chin, Carl Cerecke, Chris McCormack, Wim van Dam, Jan Dittberner, Michael John Downes, Matthias Dreier, David Dureisseix, Elliot, Hans Ehrbar, Daniel Flipo, David Frey, Hans Fugal, Robin Fairbairns, Jörg Fischer, Erik Frisk, Mic Milic Frederickx, Frank, Kasper B. Graversen, Arlo Griffiths, Alexandre Guimond, Andy Goth, Cyril Goutte, Greg Gamble, Frank Fischli, Morten Høgholm, Neil Hammond, Rasmus Borup Hansen, Joseph Hilferty, Björn Hvittfeldt, Martien Hulsen, Werner Icking, Jakob, Eric Jacoboni, Alan Jeffrey, Byron Jones, David Jones, Johannes-Maria Kaltenbach, Michael Koundouros, Andrzej Kawalec, Sander de Kievit, Alain Kessi, Christian Kern, Tobias Klauser, Jörg Knappen, Kjetil Kjernsmo, Maik Lehradt, Rémi Letot, Flori Lambrechts, Axel Liljencrantz, Johan Lundberg, Alexander Mai, Hendrik Maryns, Martin Maechler, Aleksandar S Milosevic, Henrik Mitsch, Claus Malten, Kevin Van Maren, Richard Nagy, Philipp Nagele, Lenimar Nunes de Andrade, Manuel Oetiker, Urs Oswald, Martin Pfister, Demerson Andre Polli, Nikos Pothitos, Maksym Polyakov Hubert Partl, John Refling, Mike Ressler, Brian Ripley, Young U. Ryu, Bernd Rosenlecher, Chris Rowley, Risto Saarelma, Hanspeter Schmid, Craig Schlenter, Gilles Schintgen, Baron Schwartz, Christopher Sawtell, Miles Spielberg, Geoffrey Swindale, Laszlo Szathmary, Boris Tobotras, Josef Tkadlec, Scott Veirs, Didier Verna, Fabian Wernli, Carl-Gustav Werner, David Woodhouse, Chris York, Fritz Zaucker, Rick Zaccone, and Mikhail Zotov.

4.20 中文版致謝！

中文 TEX 學會啓動的 lshort-zh-cn 修正計劃已經完工！本項計劃歷時八個月，參加的朋友有：

CTEX論壇 ID	執行章節
zpxing	前言、第二章、第五章 1-2.4 & 3、第六章
Frogge	第一章
liwenjun	第三章
lijian605	第四章
gprsnl	第五章 2.5-2.11

haginile 和 Frogge 通讀了全篇，並寫出了詳盡的勘誤表。 blackold 對於第二章亦有所貢獻。最後由 zpxing 統籌全書。

. . . . . .

原 3.20 中文版致謝！

本文檔的翻譯工作由 CTEX 版主「經典問題」倡議，歷經近十個月才得以完成。期間參與翻譯工作的朋友有：

CTEX論壇 ID	翻譯章節	源文件名
經典問題	前言	overview.tex
高原之狼	第一章	things.tex
controlong	第二章	typeset.tex
cxterm	第三章	math.tex, lssym.tex
aloft	第四章	spec.tex
ganzhi	第五章	custom.tex

在此特向這些奉獻者表示感謝!

前言

LaTeX [1] 是一種排版系統，它非常適用於生成高印刷質量的科技和數學類文檔。這個系統同樣適用於生成從簡單信件到完整書籍的所有其他種類的文檔。LaTeX 使用 TeX[2] 作爲它的格式化引擎。

這份短小的介紹描述了 LaTeX 2_ε 的使用，對 LaTeX 的大多數應用來說應該是足夠了。參考文獻 [1, 3] 對 LaTeX 系統提供了完整的描述。

這份介紹共有六章：

第一章 告訴你關於 LaTeX 2_ε 文檔的基本結構。你也會從中瞭解一點 LaTeX 的歷史。閱讀這一章後，你應該對 LaTeX 如何工作有一個大致的理解。

第二章 探究文檔排版的細節。它解釋了大部分必要的 LaTeX 命令和環境。在閱讀完這一章之後，你就能夠編寫你的第一份文檔了。

第三章 解釋了如何使用 LaTeX 排版公式。同時，大量的例子會有助於你理解 LaTeX 是如何的強大。在這個章節的結尾，你會找到列出 LaTeX 中所有可用數學符號的表格。

第四章 解釋了索引和參考文件的生成、EPS 圖形的插入。它介紹了如何使用 pdfLaTeX 生成 pdf 文檔和一些其他有用的擴展宏包。

第五章 演示如何使用 LaTeX 創建圖形。不必使用圖形軟件畫圖、存檔並插入 LaTeX 文檔，你可以直接描述圖形，然後 LaTeX 會替你畫好它。

第六章 包含一些潛在的危險信息，內容是關於如何改變 LaTeX 所產生文檔的標準佈局。它會告訴你如何把 LaTeX 的輸出變得更糟糕，或者更上一層樓，當然這取決於你的能力。

按照順序閱讀這些章節是很重要的 —— 這本書畢竟不長。一定要認眞閱讀例子，因爲在貫穿全篇的各種例子裡包含了很多的信息。

LaTeX 適用於從 PC 和 Mac 到大型的 UNIX 和 VMS 系統上。許多大學的計算機集群上安裝了 LaTeX，隨時可以使用。*Local Guide* [5] 裡應該會介紹如何使用本地安裝的 LaTeX。如果有問題，就去問給你這本小册子的人。這份文檔不會告訴你如何安裝一個 LaTeX 系統，而是教會你編寫 LaTeX 能夠處理的文檔。

如果你想取得 LaTeX 的相關材料，請訪問「Comprehensive TeX Archive Network」(CTAN) 站點，主頁是 `http://www.ctan.org`。所有的宏包也可以從 ftp 歸檔站點 `ftp://www.ctan.org` 和遍佈全球的各個鏡像站點中獲得。所有的宏包都可以在 `ftp://ctan.tug.org` 以及它遍佈全球的鏡像取得。

在本書中你會找到其他引用 CTAN 的地方，尤其是，給出你可能需要下載的軟件和文檔的指示。這裡沒有寫出完整的 url，而僅僅是其在 CTAN: 之後的樹狀結構中的位置。

請先看看 CTAN:/tex-archive/systems 中有些什麼，如果你想在自己的計算機上運行 LaTeX。

如果你有意在這份文檔中增加、刪除或者改變一些內容，請通知我。我對 LaTeX 初學者的反饋特別感興趣，尤其是關於這份介紹哪些部分很容易理解，哪些部分可能需要更好地解釋。

Tobias Oetiker <oetiker@ee.ethz.ch>

Department of Information Technology and
Electrical Engineering,
Swiss Federal Institute of Technology

這份文檔的最新版本在
CTAN:/tex-archive/info/lshort

關於這份文檔的最新中文翻譯，請諮詢
http://bbs.ctex.org

目錄

圖形清單

表格清單

Chapter 1

基礎知識

本章的第一部分給出了 LaTeX 2ε 原理及歷史的簡短介紹。第二部分集中講解 LaTeX 文檔的基本結構。讀完本章之後，你應該大致瞭解 LaTeX 的工作原理，這對你理解本書的其餘部分來說是必須的。

1.1 遊戲的名目

1.1.1 TeX

TeX 是 Donald E. Knuth 編寫的一個以排版文章及數學公式爲目標的計算機程序 [2]。1977 年，在意識到惡劣的排版質量正在影響自己的著作及文章後，Knuth 開始編寫 TeX 排版系統引擎，探索當時開始進入出版工業的數字印刷設備的潛力，尤爲希望能扭轉排版質量下滑的這一趨勢。我們現在使用的 TeX 系統發佈於 1982 年，在 1989 年又稍做改進，增加了對 8 字節字符及多語言的支持。TeX 以其卓越的穩定性、可在不同類型的電腦上運行以及幾乎沒有缺陷而著稱。TeX 的版本號不斷趨近於 π，現在爲 3.141592。

TeX 發音爲 “Tech”，其中 “ch” 和德語 “Ach”[1] 及蘇格蘭語 “Loch” 中的 “ch” 類似。“ch” 源自希臘字母，希臘文中，X 是字母 “ch” 或 “chi”。 TeX 同時也是希臘單詞 texnologia (technology) 的第一個音節。在 `ASCII` 文本環境中，TeX 寫作 `TeX`。

1.1.2 LaTeX

LaTeX 是一個宏集，它使用一個預先定義好的專業版面，可以使作者們高質量的排版和列印他們的作品。LaTeX 最初由 Leslie Lamport 編寫 [1]，它使用 TeX 程序作爲排版引擎。現在 LaTeX 由 Frank Mittelbach 負責維護。

LaTeX 的發音爲 “Lay-tech” 或 “Lah-tech”。如果在 `ASCII` 環境中引用 LaTeX，你可以輸入 `LaTeX`。 LaTeX 2ε 的發音爲 “Lay-tech two e”，在 `ASCII` 環境中寫作 `LaTeX2e`。

[1] 在德語中，“ch” 有兩種發音，有的人可能認爲 “Pech” 中較軟的 “ch” 更加合適。被問及這個問題時，Knuth 在德文 Wikipedia 中寫道： 當人們以他們喜歡的方式來拼讀 TeX 時，我並不感到生氣……在德國，更多的人喜歡較軟的 ch，因爲 X 跟在元音 e 的後面。在俄語中，‘tex’ 是一個非常普遍的單詞，讀作 ‘tyekh’。但我相信最合適的發音來自希臘語，其中 ach 和 Loch 中 ch 的發音稍尖。

1.2 基礎

1.2.1 作者、圖書設計者和排版者

出版的第一步就是作者把打好字的手稿交給出版公司，然後由圖書設計者來決定整個文檔的佈局（欄寬、字體、標題前後的間距、……）。圖書設計者會把他的排版說明寫進作者的手稿裡，再交給排版者，由排版者根據這些說明來排版全書。

一個圖書設計者要試圖理解作者寫作時的意圖。他要根據手稿的內容和他自己的職業知識來決定章節標題、文獻引用、例子及公式等等。

在一個 LaTeX 環境中，LaTeX 充當了圖書設計者的角色，而 TeX 則是其排版者。但是 LaTeX「僅僅」是一個程序，因此它需要很多的指導。作者必須提供額外的信息，來描述其著作的邏輯結構。這些信息是以「LaTeX 命令」的形式寫入文檔中的。

這和大多數現代文字處理工具，如 *MS Word* 及 *Corel WordPerfect* 所採用的所見即所得 (WYSIWYG[2]) 的方式有很大區別。使用這些工具時，作者在向計算機中輸入文檔的同時，通過互動的方式確定文章的佈局。作者可以從螢幕上看到作品的最終列印效果。

而使用 LaTeX時，一般是不能在輸入文檔的同時看到最終的輸出效果的，但是使用 LaTeX處理文檔之後，便可以在螢幕上預覽最終的輸出效果。因此在眞正列印文檔之前還是可以做出改正的。

1.2.2 版面設計

排版設計是一門工藝。不熟練的作者認爲書籍設計僅僅是個美學問題，因而經常會犯嚴重的格式錯誤 ——「如果一份文檔從藝術的角度看起來不錯，那麼它的設計就是成功的」。不過作爲一份用來閱讀而不是掛在畫廊裡的文檔，可讀性和可理解性遠比漂亮的外觀重要。例如：

- 必須選定字號和標題的序號，使讀者能清楚的理解章節的結構。
- 每一行既要足夠短以避免讀者眼睛疲勞，又要足夠長以維持頁面的美觀。

在使用所見即所得系統 (WYSIWYG) 時，作者經常會寫出一些看上去漂亮，但結構欠清晰或不連貫的文章來。LaTeX 通過強制作者聲明文檔的**邏輯**結構，來避免這些排版格式錯誤。然後，LaTeX 再根據文檔的結構選擇最合適的版面格式。

1.2.3 優勢和不足

使用所見即所得 (WYSIWYG) 的人和使用 LaTeX 的人遇到一起時，他們經常討論的話題就是「相比一般文字處理軟件，LaTeX 的優勢 (advantages of LaTeX)」或者不足。當這樣的討論開始時，你最好保持低調，因爲討論往往會失控。但有時你也不能逃避……

下面便是一些武器。LaTeX 優於一般文字處理軟件之處可歸納如下：

- 提供專業的版面設計，可以使一份文檔看起來就像「印刷品」一樣。
- 可以方便的排版數學公式。

[2] What you see is what you get.

- 用戶只需要學一些聲明文檔邏輯結構的簡單易懂的命令，而不必對文檔的實際版面修修補補。
- 可以容易的生成像脚註、引用、目錄和參考文獻等很多複雜的結構。
- 很多不被基本 LaTeX支持的排版工作，可以由添加免費的宏包來完成。例如，支持在文件中插入 PostScript 格式圖像的宏包及排版符合各類準確標準的參考文獻的宏包等。很多這類宏包在 *The LaTeX Companion* [3]中都有說明。
- LaTeX 鼓勵作者按照合理的結構寫作，因爲 LaTeX 就是通過指明文檔結構來進行排版工作的。
- TeX，作爲 LaTeX 2_ε的排版引擎，不僅免費，而且具有很高的可移植性，幾乎可以在任何硬件平台上運行。

LaTeX 也有一些不足之處。儘管我可以確定別人可以列出幾百條，我自己卻很難找到一些比較理智的 ;-)

- 沒有原則的人不能使用 LaTeX 很好地工作……
- 儘管可以調節預先定義好的文檔版面佈局中的一些參數，但設計一個全新的版面還是很睏難的，並會耗費大量時間[3]。
- 很難用 LaTeX 來寫結構不明、組織無序的文檔。
- 即使有一個令人鼓舞的開端，你也可能無法完全掌握其精髓。

1.3 LaTeX 源文件

LaTeX 源文件爲普通的 ASCII 文件，你可以使用任何文本編輯器來創建。LaTeX 源文件不僅包含了要排版的文本，而且也包含了告訴 LaTeX 如何排版這些文本內容的命令。

1.3.1 空白距離

空格和製表符等空白字符在 LaTeX 中被看作相同的空白距離 (space)。**多**個連續的空白字符等同於**一**個空白字符。在句首的空白距離一般會被忽略，單個空行也被認爲是一個「空白距離」。

兩行文本間的空白行標誌著上段的結束和下段的開始。**多**個空白行的作用等同於**一**個空白行。下面便是一個例子，左邊是源文件中的文本，右邊是排版後的結果。

```
It does not matter whether you
enter one or several     spaces
after a word.

An empty line starts a new
paragraph.
```

It does not matter whether you enter one or several spaces after a word.

An empty line starts a new paragraph.

[3]傳聞這將是未來的 LaTeX3 系統中的一個重要組成部分。

1.3.2 特殊字符

下面的這些字符是 LaTeX 中的保留字符 (reserved characters)，它們或在 LaTeX 中有特殊的意義，或不一定存在於所有字庫中。如果你直接在文本中輸入這些字符，通常它們不會被輸出，而且還會導致 LaTeX 做一些你不希望發生的事情。

```
# $ % ^ & _ { } \
```

如你看到的，在這些字符前加上反斜線，它們就可以正常的輸出到文檔中。

```
\# \$ \% \^{} \& \_ \{ \} \ {}
```

$ % ^ & _ { }

其他一些特殊符號可以由數學環境中的特殊命令或重音命令得到。反斜線 \ 不能通過在其前面加另一個反斜線得到 (\\)；這是一個用來換行的命令[4]。

1.3.3 LaTeX 命令

LaTeX 命令 (commands) 是大小寫敏感的，有以下兩種格式：

- 以一個反斜線 (backslash) \ 開始，命令名只由字母組成。命令名後的空格符、數字或任何非字母的字符都標誌著該命令的結束。
- 由一個反斜線和非字母的字符組成。

LaTeX 忽略命令之後的空白字符。如果你希望在命令後得到一個空格，可以在命令後加上 {} 和一個空格，或加上一個特殊的空格命令。{} 將阻止 LaTeX 吃掉命令後的所有空格。

```
I read that Knuth divides the
people working with \TeX{} into
\TeX{}nicians and \TeX perts.\\
Today is \today.
```

I read that Knuth divides the people working with TeX into TeXnicians and TeXperts.
Today is September 15, 2008.

有些命令需要一個參數 (parameter)，該參數用花括號 (curly braces) { } 括住並寫在命令的後面。一些命令支持可選參數 (optional parameters)，可選參數可用方括號 (square brackets) [ ] 括住，然後寫在命令的後面。下面的例子中使用了一些 LaTeX 命令，不要著急，後面將解釋它們的含義。

```
You can \textsl{lean} on me!
```

You can *lean* on me!

```
Please, start a new line
right here!\newline
Thank you!
```

Please, start a new line right here!
Thank you!

[4] 試試 $\backslash$ 命令，它將生成一個 '\'。

1.3.4 註釋

當 LaTeX 處理一個源文件時，如果遇到一個百分號 %，LaTeX 將忽略 % 後的該行內容，換行符以及下一行前的空白字符。

我們可以據此在源文件中寫一些註釋，而且這些註釋並不會出現在最後的排版結果中。

```
This is an % stupid
% Better: instructive <----
example: Supercal%
              ifragilist%
    icexpialidocious
```

This is an example: Supercalifragilisticexpialidocious

符號 % 也可以用來斷開不能含有空白字符或換行符的較長輸入內容。

如果註釋的內容較長，你可以使用 verbatim 宏包提供的 comment 環境。當然，在使用該環境前，你要在文檔的導言區（後面將會解釋其含義）加上命令 \usepackage{verbatim}。

```
This is another
\begin{comment}
rather stupid,
but helpful
\end{comment}
example for embedding
comments in your document.
```

This is another example for embedding comments in your document.

需要注意的是以上做法在數學環境等複雜環境中不起作用。

1.4 源文件的結構

當 LaTeX 2ε 處理源文件時，它希望源文件遵從一定的結構 (structure)。因此，每個源文件都要以如下命令開始

```
\documentclass{...}
```

這條命令指明了你所寫的源文檔的類型。然後，你就可以加入控制整篇文檔樣式的命令，或者載入一些爲 LaTeX 增加新特性的宏包 (package)。可以用如下命令載入一個宏包

```
\usepackage{...}
```

當完成所有的設置工作後[5]，你可以用下面的命令開始文檔的主體

```
\begin{document}
```

現在你就可以輸入帶有 LaTeX 命令的正文了。在文章末尾使用命令

```
\end{document}
```

來告訴 LaTeX 文檔已經結束。LaTeX 會忽略此命令後的所有內容。

圖 1.1 顯示的是一個簡單的 LaTeX 2ε 文檔的結構。一個較爲複雜的源文件 (input file) 結構如圖 1.2 所示。

[5] 在 \documentclass 和 \begin{document} 之間的部分稱作**導言區** (preamble)。

```
\documentclass{article}
\begin{document}
Small is beautiful.
\end{document}
```

圖 1.1 – 一個簡單的 LaTeX 源文件。

```
\documentclass[a4paper,11pt]{article}
% define the title
\author{H. Partl}
\title{Minimalism}
\begin{document}
% generates the title
\maketitle
% insert the table of contents
\tableofcontents
\section{Some Interesting Words}
Well, and here begins my lovely article.
\section{Good Bye World}
\ldots{} and here it ends.
\end{document}
```

圖 1.2 – article 類 LaTeX 源文件例子，該例中的所有命令後面都會講到。

1.5 一個典型的命令行過程

我敢打賭你現在一定非常渴望嘗試第 6 頁上短小簡潔的 LaTeX 源文件。下面便是一些幫助：LaTeX 本身沒有圖形用戶界面或漂亮的按鈕，它僅僅是一個處理你提供的源文件的程序。有些 LaTeX 安裝版本提供了一個前端圖形界面，你可以通過點擊按鈕來編譯你的源文件。其他的一些系統上可能就要使用命令來編譯源文件，下面演示的就是如何在一個基於文本的系統上讓 LaTeX 編譯你的源文件。需要注意：以下演示的前提是 LaTeX 已經正確的安裝到了你的電腦中[6]。

1. 創建並編輯你的源文件。源文件必須是普通的 ASCII 格式。在 Unix 系統下，所有的編輯器都可以創建這樣的文件。在 Windows 系統下，你必須確保文件以 ASCII 或**普通文本**格式保存。當選取你源文件的文件名時，確保它的擴展名是 `.tex`。

2. 運行 LaTeX 編譯你的源文件。如果成功的話，你將會得到一個 `.dvi` 文件。爲了得到目錄和所有的內部引用，可能要多次運行 LaTeX。當源文件中存在錯誤時，LaTeX 會告訴你錯誤並停止處理源文件。輸入 `ctrl-D` 可以返回到命令行。

   ```
   latex foo.tex
   ```

3. 現在可以通過幾種方法來預覽得到的 DVI 文件。你可以使用下列命令將文件顯示到螢幕上

   ```
   xdvi foo.dvi &
   ```

   這種方法只適用於安裝了 X11 的 Unix 系統。如果你使用的是 Windows 系統，可以使用 `yap` 來預覽（或其他預覽程序）。

   你也可以使用 Ghostscript 將 dvi 文件轉換成 PostScript 文件來列印或預覽。

   ```
   dvips -Pcmz foo.dvi -o foo.ps
   ```

   如果你的 LaTeX 系統中帶有 `dvipdf` 工具的話，就可以直接將 `.dvi` 文件轉換成 pdf 文件。

   ```
   dvipdf foo.dvi
   ```

1.6 文檔佈局

1.6.1 文檔類

當 LaTeX 處理源文件時，首先需要知道的就是作者所要創建的文檔類型。文檔類型可由 `\documentclass` 命令來指定。

```
\documentclass[options]{class}
```

class 指定想要的文檔類型。表 1.1 給出了一些文檔類型的解釋。LaTeX 2_ε 發行版中還提供了其他一些文檔類，像信件和幻燈片等。通過 *options* 參數可以定

[6]這是在大部分 Unix 系統下的情況……高手使用 Unix，所以…… ;-)

製文檔類的屬性。不同的選項之間須用逗號隔開。標準文檔類的最常用選項如表 1.2 所示。

例子：一個 LaTeX 源文件以下面一行開始

```
\documentclass[11pt,twoside,a4paper]{article}
```

這條命令會引導 LaTeX 使用 *article* 格式、**11 磅大小的字體**來排版該文檔，並得到在 *A4* 紙上**雙面列印**的效果。

1.6.2 宏包

排版文檔時，你可能會發現某些時候基本的 LaTeX 並不能解決你的問題。如果想插入圖形 (graphics)、彩色文本 (coloured text) 或源代碼到你的文檔中，你就需要使用宏包來增強 LaTeX 的功能。可使用如下命令調用宏包

`\usepackage[`*options*`]{`*package*`}`

這裡 *package* 是宏包的名稱，*options* 是用來啓動宏包特殊功能的一組關鍵詞。很多宏包隨 LaTeX 基本發行版一起發佈 (見表 1.3)，其他的則單獨發佈。你可以在所安裝的 LaTeX 系統中找到更多的宏包相關信息。*The LaTeX Companion* [3]提供了關於宏包的重要信息，它包含了數百個宏包的描述及如何寫作自己的 LaTeX 2_ε 擴展的信息。

現代的 TeX發行版包含了大量免費的宏包。如果你使用的是 Unix 系統，可以使用命令 `texdoc` 搜索宏包的說明文檔。

1.6.3 頁面樣式

LaTeX 支持三種預定義的頁眉/頁脚 (header/footer) 樣式，稱爲頁面樣式 (page style)。如下命令

`\pagestyle{`*style*`}`

中的 *style* 參數確定了使用哪一種頁面樣式。表 1.4 列出了預定義的頁面樣式。

表 1.1 – 文檔類。

`article`	排版科學期刊、演示文檔、短報告、程序文檔、邀請函… …
`proc`	一個基於 article 的會議文集類。
`minimal`	非常小的文檔類。只設置了頁面尺寸和基本字體。主要用來查錯。
`report`	排版多章節長報告、短篇書籍、博士論文… …
`book`	排版書籍。
`slides`	排版幻燈片。該文檔類使用大號 sans serif 字體。也可以選用 FoilTeX[a] 來得到相同的效果。

[a] `macros/latex/contrib/supported/foiltex`

表 1.2 – 文檔類選項。

`10pt`, `11pt`, `12pt`　設置文檔中所使用的字體的大小。如果該項沒有指定，默認使用 `10pt` 字體。

`a4paper`, `letterpaper`, ...　定義紙張的尺寸。缺省設置爲 `letterpaper`。此外，還可以使用 `a5paper`, `b5paper`, `executivepaper` 以及 `legalpaper`。

`fleqn`　設置行間公式爲左對齊，而不是居中對齊。

`leqno`　設置行間公式的編號爲左對齊，而不是右對齊。

`titlepage`, `notitlepage`　指定是否在文檔標題 (document title) 後另起一頁。`article` 文檔類缺省設置爲不開始新頁，`report` 和 `book` 類則相反。

`onecolumn`, `twocolumn`　LaTeX 以單欄 (one column) 或雙欄 (two column) 的方式來排版文檔。

`twoside`, `oneside`　指定文檔爲雙面或單面列印格式。`article` 和 `report` 類爲單面 (single sided) 格式，`book` 類缺省爲雙面 (double sided) 格式。注意該選項只是作用於文檔樣式，而**不會**通知列印機以雙面格式列印文檔。

`landscape`　將文檔的列印輸出佈局設置爲 landscape 模式。

`openright`, `openany`　決定新的一章僅在奇數頁開始還是在下一頁開始。在文檔類型爲 `article` 時該選項不起作用，因爲該類中沒有定義「章」(chapter)。`report` 類默認在下一頁開始新一章而 `book` 類的新一章總是在奇數頁開始。

表 1.3 – 隨 LaTeX一起發行的宏包。

doc 排版 LaTeX 的說明文檔。具體描述見 `doc.dtx`[a] 及 *The LaTeX Companion* [3]。

exscale 提供了按比例伸縮的數學擴展字體。
具體描述見 `ltexscale.dtx`。

fontenc 指明使用哪種 LaTeX 字體編碼 (font encoding)。
具體描述見 `ltoutenc.dtx`。

ifthen 提供如下形式的命令
'if ... then do ... otherwise do ....'
具體描述見 `ifthen.dtx` 及 *The LaTeX Companion* [3]。

latexsym 提供 LaTeX 符號字體。具體描述見 `latexsym.dtx` 及 *The LaTeX Companion* [3]。

makeidx 提供排版索引的命令。具體描述見第 4.3 節及 *The LaTeX Companion* [3]。

syntonly 編譯文檔而不生成 dvi 文件（常用於查錯）。

inputenc 指明使用哪種輸入編碼，如 ASCII, ISO Latin-1, ISO Latin-2, 437/850 IBM code pages, Apple Macintosh, Next, ANSI-Windows 或用戶自定義編碼。具體描述見 `inputenc.dtx`。

[a] 你的系統中應該安裝了該文件，輸入命令 `latex doc.dtx` 處理該文件可得到一個 `dvi` 文件。類似的方法適用於本表格中的其他 `.dtx` 文件。

表 1.4 – LaTeX預定義的頁面樣式。

`plain` 在頁脚正中顯示頁碼。這是頁面樣式的缺省設置。

`headings` 在頁眉中顯示章節名及頁碼，頁脚空白。（本文即採用此樣式）

`empty` 將頁眉頁脚都設爲空白。

可以通過如下命令來改變當前頁面的頁面樣式

`\thispagestyle{`*style*`}`

如何創建自定義頁眉頁腳的說明可以參見 *The LaTeX Companion* [3] 及第 58 頁的第 4.4 節。

1.7 各類 LaTeX 文件

使用 LaTeX 時，你可能很快發現自己置身於各種不同擴展名 (extension) 或毫無線索的文件形成的迷宮之中。下面的列表解釋了在使用 LaTeX 時可能遇到的文件類型。要注意的是，下表不是所有的擴展名列表，如果你發現有重要的文件類型沒有收錄進來，請通知我。

`.tex` LaTeX 或 TeX 源文件。可以使用 `latex` 命令編譯。

`.sty` LaTeX 宏包文件。可以使用 `\usepackage` 命令將宏包文件載入到你的 LaTeX 文檔中。

`.dtx` 文檔化 TeX 文件。這是 LaTeX 宏包文件的主要發佈格式。如果編譯 `.dtx` 文檔，將會得到其中包含的 LaTeX 宏包文件的文檔化宏代碼。

`.ins` 對應 `.dtx` 文件的安裝文件。如果你從網上下載了一個 LaTeX 的宏包文件，其中一般會包含一個 `.dtx` 文件和一個 `.ins` 文件。使用 LaTeX 處理 `.ins` 文件可以解開 `.dtx` 文件。

`.cls` 定義文檔外觀形式的類文件，可以通過使用 `\documentclass` 命令選取。

`.fd` 字體描述文件，可以告訴 LaTeX 有關新字體的信息。

下面這些文件是使用 LaTeX 處理源文件時產生的：

`.dvi` 設備無關文件。這是運行 LaTeX 編譯的主要結果。你可以使用 DVI 預覽器預覽其內容或使用 `dvips` 或其他程序輸出到列印機。

`.log` 記錄了上次編譯時的詳細信息。

`.toc` 儲存了所有的章節標題。下次編譯時將讀取該文件並生成目錄。

`.lof` 和 `.toc` 文件類似，可生成圖形目錄。

`.lot` 和 `.toc` 文件類似，可生成表格目錄。

`.aux` 用來向下次編譯傳遞信息的輔助文件。主要儲存交叉引用的相關信息。

`.idx` 如果文檔中包含索引，LaTeX 將使用該文件存儲所有的索引詞條。此文件需要使用 `makeindex` 處理，詳見位於 57 頁的第 4.3 節。

`.ind` 處理過的 `.idx` 文件。下次編譯時將讀入到你的文檔中。

`.ilg` 和 `.log` 文件類似，記錄了 `makeindex` 命令運行的詳細信息。

1.8 大型項目

當處理大型文檔時，最好將文檔分割成爲幾部分。LaTeX 有兩個命令可以幫助你完成這項工作。

`\include{`*filename*`}`

你可以使用該命令將名爲 *filename.tex* 的文檔內容插入到當前文檔中。需要注意的是，在處理插入的 *filename.tex* 文檔前，LaTeX 會另起一頁。

第二個命令只能在導言區使用。它可以讓 LaTeX 僅讀入某些 `\include` 文件。

`\includeonly{`*filename*`,`*filename*`,...}`

這條命令在文檔的導言區執行後，在所有的 `\include` 命令中，只有文檔名出現在 `\includeonly` 的命令參數中的文檔才會被導入。注意文檔名和逗號之間不能有空格。

`\include` 命令會在新的一頁上排版載入的文本。當使用 `\includeonly` 命令時會很有幫助，因爲即使一些載入的文本被忽略，分頁處也不會發生變化。有些時候可能不希望在新的一頁上排版載入的文本，這時可以使用命令

`\input{`*filename*`}`

`\input` 命令只是簡單的載入指定的文本，沒有其他限制。

如果想讓 LaTeX 快速的檢查文檔中的錯誤，可以使用 syntonly 宏包。它可以使 LaTeX 瀏覽整個文檔，檢查語法錯誤和使用的命令，但並不生成 DVI 輸出。在這種模式下，LaTeX 運行速度很快，可以爲你節省寶貴的時間。syntonly 宏包的使用非常簡單：

```
\usepackage{syntonly}
\syntaxonly
```

如果想產生分頁，只要註釋掉第二行即可 (在前面加上一個百分號 %)。

Chapter 2

文本排版

閱讀了前一章之後，你應該瞭解關於如何創建一個 LaTeX 文檔的基本知識了。在這一章裡，我將補充其餘部分，使你能夠生成實際文檔。

2.1 文本和語言結構

By Hanspeter Schmid <hanspi@schmid-werren.ch>

書寫文本的主旨是（某些現代 DAAC[1] 文化除外），向讀者傳遞觀點、信息或者知識。如果這些觀點被很好地組織起來，那麼讀者會得到更好的理解。而且，如果排版形式反映內容的邏輯和語義結構，讀者就能看到也更喜歡文章的這種脈絡。

LaTeX 不同於其它排版系統之處在於，你必須告訴它文本的邏輯和語義結構。然後它根據類文件和各種樣式文件中給定的「規則」生成相應格式的文本。

LaTeX 最重要的文本單元（印刷術上的）是段落 (paragraph)。我們稱段落爲「文本單元」，因爲段落是連續思想或者觀點在排版上的反映。在下一節裡，你將學會在源代碼中如何使用 \\ 來強迫換行，如何使用空行來分段。因此，一旦開始表達新的思想，就應該另起一段，否則換行就夠了。如果無法決定是否分段，想像一下你的文字是觀點和思想的載體。如果分段後，原來的思想仍在繼續，就應該取消分段。如果有些行在同一段落裡闡述了新的思想，那麼應該分段。

大部分人完全低估了恰當分段的重要性。許多人甚至不知道分段表示什麼，或者，特別是在 LaTeX裡，設置了分段但卻渾然不知。後一錯誤特別容易發生在文本中使用公式的情況。觀察下面的例子並理解爲什麼有時公式前後都使用空行（分段），而有時不這樣。（如果你還不能掌握裡面所用的命令以至於無法理解這些例子，請在閱讀這一章和下一章後再閱讀這一節。）

```
% Example 1
\ldots when Einstein introduced his formula
\begin{equation}
  e = m \cdot c^2 \; ,
\end{equation}
which is at the same time the most widely known
and the least well understood physical formula.
```

[1] 爲標新立異而不講成本，譯自 the Swiss German UVA (Um's Verrecken Anders).

```
% Example 2
\ldots from which follows Kirchhoff's current law:
\begin{equation}
  \sum_{k=1}^{n} I_k = 0 \; .
\end{equation}

Kirchhoff's voltage law can be derived \ldots


% Example 3
\ldots which has several advantages.

\begin{equation}
  I_D = I_F - I_R
\end{equation}
is the core of a very different transistor model. \ldots
```

另一個更小的文本單元是句子。在英文文本中，結束句子的句點後面的空格比縮略詞的句點後面的空格更長。LaTeX 試圖判斷你需要哪一個，如果 LaTeX 判斷錯了，你必須告訴它你需要什麼。這將會在下一章裡談到。

文本的結構甚至還包括句子的成份。大部分語言的標點規則非常複雜，但在許多語言（包括德文和英文）中，如果你記住逗號的意思：在語流中的短暫停頓，那麼幾乎所有的逗號都不會被用錯。如果你不確定在什麼地方應該使用逗號，大聲地朗讀句子並在每一個逗號處喘口氣。在呼吸彆扭的地方刪除逗號，而在需要喘口氣（或者需要短暫停頓）的地方插入一個逗號。

最後，通過包含段落的章、節和子節等等，段落應該在更高層次被有邏輯地組織起來。然而，使用諸如 `\section{The Structure of Text and Language}` 的排版效果，是如此明顯以至於如何使用這些高層次的結構是不言而喻的。

2.2 斷行和分頁

2.2.1 對齊段落

通常書籍是用等長的行來排版的。爲了優化整個段落的內容，LaTeX 在單詞之間插入必要的斷行點 (line break) 和間隙。如果一行的單詞排不下，LaTeX 也會進行必要的斷詞。段落如何排版依賴於文檔類別。通常，每一段的第一行有縮進，在兩段之間沒有額外的間隔。更多的內容請參考第 6.3.2 節。

在特殊情形下，有必要命令 LaTeX 斷行

`\\ or \newline`

另起一行，而不另起一段。

`\\*`

在強制斷行後，還禁止分頁。

`\newpage`

另起一頁。

`\linebreak[`n`], \nolinebreak[`n`], \pagebreak[`n`], \nopagebreak[`n`]`

上述命令的效果可以從它們的名稱看出來。通過可選參量 n，作者可以影響這些命令的效果。n 可以取爲 0 和 4 之間的數。如果命令的效果看起來非常差，把 n 取爲小於 4 的數，可以讓 LaTeX 在排版效果不佳的時候選擇忽略這個命令。不要把這些 “break” 命令與 “new” 命令混淆。即使你給出了 “break” 命令，LaTeX 仍然試圖對齊頁面的右邊界。如果你眞想另起一行，就使用相應的命令。猜猜該是什麼命令！

LaTeX 總是儘可能產生最好的斷行效果。如果斷行無法達到 LaTeX 的高標準，就讓這一行在段落的右側溢出。然後在處理源文件的同時，報告溢出的消息 (“overfull hbox”)。這最有可能發生在 LaTeX 找不到合適的地方斷詞的時候[2]。你可以使用 `\sloppy` 命令，告訴 LaTeX 降低一點兒標準。它通過增加單詞之間的間隔，以防止出現過長的行，雖然最終的輸出結果不是最優的。在這種情況下給出警告 (“underfull hbox”)。在大多數情況下得到的結果看起來不會非常好。 `\fussy` 命令把 LaTeX 恢復爲缺省狀態。

2.2.2 斷詞

必要時 LaTeX 就會斷詞。如果斷詞算法不能確定正確的斷詞點，可以使用如下命令告訴 TeX 如何彌補這個缺憾。

命令

`\hyphenation{`*word list*`}`

使列於參量中的單詞僅在注有 “-” 的地方斷詞。命令的參量僅由正常字母構成的單詞，或由 LaTeX 視爲正常字母的符號組成。當斷詞命令出現時，根據正在使用的語言，斷詞的提示就已經被存好待選了。這意味著如果你在文檔導言中設置了斷詞命令，它將影響英文的斷詞。如果斷詞命令置於 `\begin{document}` 後面，而且你正使用比方 babel 的國際語言支持宏包，那麼斷詞提示在由 babel 啓動的語言中就處於活動狀態。

下面的例子允許對 “hyphenation” 和 “Hyphenation” 進行斷詞，卻根本不允許 “FORTRAN”, “Fortran” 和 “fortran” 進行斷詞。在參量中不允許出現特殊的字符和符號。

例子：

```
\hyphenation{FORTRAN Hy-phen-a-tion}
```

[2]當發生 (Overfull hbox) 時，雖然 LaTeX 給出一個警告並顯示溢出的那一行，但是不太容易發現溢出的行。如果你在 `\documentclass` 命令中使用選項 `draft`， LaTeX 就在溢出行的右邊標以粗黑線。

命令 \- 在單詞中插入一個自主的斷詞點。它也就成爲這個單詞中允許出現的唯一斷詞點。對於包含特殊字符（例如：注音字符）的單詞，這個命令是特別有用的，因爲對於他們，LaTeX 不會自動斷詞[3]。

```
I think this is: su\-per\-cal\-%
i\-frag\-i\-lis\-tic\-ex\-pi\-%
al\-i\-do\-cious
```

I think this is: supercalifragilisticexpialidocious

命令

`\mbox{`*text*`}`

保證把幾個單詞排在同一行上。在任何情況下，這個命令把它的參量排在一起。

```
My phone number will change soon.
It will be \mbox{0116 291 2319}.

The parameter
\mbox{\emph{filename}} should
contain the name of the file.
```

My phone number will change soon. It will be 0116 291 2319.

The parameter *filename* should contain the name of the file.

命令 \fbox 和 \mbox 類似，此外它還能圍繞內容畫一個框。

2.3 内置字符串

在前面的例子中，你已經看到用來排版特殊文本字符串的一些非常簡單的 LaTeX 命令了。

命令	例子	描述
\today	September 15, 2008	今日日期
\TeX	TeX	你最喜愛的排版工具
\LaTeX	LaTeX	遊戲的名目
\LaTeXe	LaTeX 2_ε	現在的化身

2.4 特殊字符和符號

2.4.1 引號

你不能再像在打字機上那樣，把 " 用作引號 (quotation marks)。在印刷中有專門的左引號和右引號。在 LaTeX 中，用兩個 `（重音）產生左引號，用兩個 '（直立引號）產生右引號。一個 ‘ 和一個 ’ 產生一個單引號。

```
``Please press the `x' key.''
```

“Please press the ‘x’ key.”

當然我知道這種實現機制不是最理想的，無論字體如何，它總是一個反向的勾號或者重音符 (`) 當左引號，直立引號 (') 當右引號。

[3]除非你正在使用新的 DC 字體 (DC font)。

2.4.2 破折號和連字號

LaTeX 中有四種短劃 (dash) 標點符號。連續用不同數目的短劃，可以得到其中的三種。第四個實際不是標點符號，它是數學中的減號：

```
daughter-in-law, X-rated\\
pages 13--67\\
yes---or no? \\
$0$, $1$ and $-1$
```

daughter-in-law, X-rated
pages 13–67
yes—or no?
0, 1 and −1

這些短劃線是: '-' 連字號 (hyphen)，'–' 短破折號 (en-dash)，'—' 長破折號 (em-dash) 和 '−' 減號 (minus sign)。

2.4.3 波浪號(∼)

波浪號經常和網址用在一起。它在 LaTeX 中，可用 \~ 產生，但其結果：˜ 卻不是你眞正想要的。試一下這個：

```
http://www.rich.edu/\~{}bush \\
http://www.clever.edu/$\sim$demo
```

http://www.rich.edu/˜bush
http://www.clever.edu/∼demo

2.4.4 度的符號 (○)

下面的例子演示了在 LaTeX 中如何排版度的符號 (degree symbol)：

```
It's $-30\,^{\circ}\mathrm{C}$.
I will soon start to
super-conduct.
```

It's −30 °C. I will soon start to super-conduct.

textcomp 宏包裡有另外一個度的符號 `\textcelsius`。

2.4.5 歐元符號 (€)

現在撰寫有關貨幣的文章，通常需要歐元符號。現有的許多字體都包含它。在你的導言區載入 textcomp 宏包，

```
\usepackage{textcomp}
```

你就可以使用命令

```
\texteuro
```

來生成歐元符號。
如果你的字體不提供或者你不喜歡它給出的歐元符號，還有兩個選擇：
首先是 eurosym 宏包。它提供了官方的歐元符號：

```
\usepackage[official]{eurosym}
```

如果你希望得到跟所用字體匹配的歐元符號，使用選項 `gen` 替換 `official`。

marvosym 宏包也提供了很多符號，包括一個名爲 `\EURtm` 的歐元符號。它的缺點是沒有提供歐元符號的斜體 (slanted) 和粗體 (bold) 變形。

表 2.1 – 歐元符號工具箱。

LM+textcomp	`\texteuro`	€	€	€
eurosym	`\euro`	€	€	€
[gen]eurosym	`\euro`	€	€	€
marvosym	`\EURtm`	€	€	€

2.4.6 省略號 (...)

在打字機上，逗號 (comma) 或句號 (period) 佔據的空間和其他字母相等。在書籍印刷中，這些字符僅佔據一點兒空間，並且與前一個字母貼得非常緊。所以不能只鍵入三個點來輸出「省略號」(ellipsis)，因爲間隔劃分得不對。有一個專門的命令輸出省略號。它被稱爲

`\ldots`

```
Not like this ... but like
this:\\ New York, Tokyo,
Budapest, \ldots
```

Not like this ... but like this:
New York, Tokyo, Budapest, ...

2.4.7 連字

一些字母組合不是簡單鍵入一個個字母得到得的，而實際上用到了一些特殊符號。

效果應爲 ff fi fl ffi... 而不是 ff fi fl ffi ...

這就是所謂的連字 (ligature)，在兩個字母之間插入一個 `\mbox{}`，可以禁止連字。對於由兩個詞構成的單詞，這可能是必要的。

```
Not shelfful\\
but shelf\mbox{}ful
```

Not shelfful
but shelfful

2.4.8 注音符號和特殊字符

LaTeX 支持來自許多語言中的注音符號 (accent) 和特殊字符 (special character)。表 2.2 就字母 o 列出了所有的注音符號。對於其他字母也自然有效。

在字母 i 和 j 上標一個注音符號，它的點兒必須去掉。這個可由 `\i` 和 `\j` 做到。

```
H\^otel, na\"\i ve, \'el\`eve,\\
sm\o rrebr\o d, !`Se\ norita!,\\
Sch\"onbrunner Schlo\ss{}
Stra\ss e
```

Hôtel, naïve, élève,
smørrebrød, ¡Se norita!,
Schönbrunner Schloß Straße

表 2.2 – 注音符號和特殊字符。

ò	\`o	ó	\'o	ô	\^o	o	\ o
ō	\=o	ȯ	\.o	ö	\"o	ç	\c c
ǒ	\u o	ǒ	\v o	ő	\H o	ǫ	\c o
ọ	\d o	o͟	\b o	o͡o	\t oo		
œ	\oe	Œ	\OE	æ	\ae	Æ	\AE
å	\aa	Å	\AA				
ø	\o	Ø	\O	l	\l	L	\L
ı	\i	ȷ	\j	¡	!`	¿	?`

2.5 國際語言支持

如果你需要用英文以外的語文 (language) 書寫文件，LaTeX 有兩個地方必須設定好：

1. 所有自動生成的字符串[4]必須適用於新語言。對於許多種語言，這個任務可由 Johannes Braams 編的宏包 babel 完成。
2. 對於一種新語言，LaTeX 需要知道它的斷詞規則。將斷詞規則輸入 LaTeX 有些難度。這是說爲不同斷詞模式重建格式文件是行得通的。對此 *Local Guide* [5] 給了更多的信息。
3. 特定語言的排版規則。比如法語中，每一個冒號 (:) 前面必須留出一定的空白。

如果你的系統已經設定好了，你可以通過在命令 `\documentclass` 後添加命令

`\usepackage[`*language*`]{babel}`

來啓動宏包 babel。已經被你的 LaTeX 系統支持的語言列表會在每次編譯的時候顯示。對於選定的語言，宏包 babel 將自動啓動適當的斷詞規則。如果 LaTeX 的格式文件不支持在所選擇的語言中斷詞，除了失去斷詞功能，宏包 babel 仍起作用，當然這對於排版效果有很大的負面影響。

對於很多種語言，宏包 babel 也提供專門的新命令來簡化特殊字符的輸入。例如德文 (German) 包含很多元音變音 (äöü) 。利用 babel，你能用 `"o` 而不是 `\"o` 來輸入 ö。

[4] 目錄、圖形清單 … …

如果爲 babel 指定了多種語言

```
\usepackage[languageA,languageB]{babel}
```

選項中的最後一種語言會被啓動（即 languageB）。你可以使用

```
\selectlanguage{languageA}
```

來改變被啓動的語言。

大多數現代的計算機系統允許直接從鍵盤輸入某國的字母。爲了處理大量不同語系以及/或者計算機平台使用的輸入編碼，LaTeX 使用 inputenc 宏包：

```
\usepackage[encoding]{inputenc}
```

當使用這個宏包時，應該考慮其他人可能因爲使用不同的編碼，在其計算機上或許不能顯示你的源文件。例如，德語元音變音 ä 的編碼爲 132，在一些使用 ISO-LATIN 1 的 Unix 系統上，它的編碼就成了 228；但是 Windows 上的 Cyrillic 編碼 cp1251 裡卻根本沒有這個字母。所以應小心使用這個功能。根據你使用的系統類型，下列編碼可能會派得上用場 [5]。

Operating system	encodings	
	western Latin	Cyrillic
Mac	`applemac`	`macukr`
Unix	`latin1`	`koi8-ru`
Windows	`ansinew`	`cp1251`
DOS, OS/2	`cp850`	`cp866nav`

如果你有一份多語言文檔，其中的編碼會有衝突。這時可以使用 ucs 宏包來選擇 unicode。

```
\usepackage{ucs}
\usepackage[utf8x]{inputenc}
```

會讓你創建的 LaTeX 文檔使用 `utf8x`，它是一種多字節的編碼，其中每個字符需要最少一個字節，最多 4 個字節。

字體編碼是另外一個問題。它定義於一種 TeX 字體裡每個字母的存放位置。幾種不同的輸入編碼可以被映射到一種字體編碼，這樣減少了所需的字體集數量。字體編碼通過 fontenc 宏包來處理：

```
\usepackage[encoding]{fontenc}
```

其中 *encoding* 是字體編碼。可以同時載入幾種編碼。

默認的 LaTeX 字體編碼是 `OT1`，Computer Modern TeX 字體的原有編碼。它只包含了 7-bit ASCII 字符集的 128 個字符。需要注音字符的時候， TeX 把一個正常的字符附上重音符來創建它。雖然輸出結果看上去很完美，但這種方法停止了對注音字符的自動斷詞功能。另外，這種方法不能創建一些拉丁字母，而且對非拉丁字母一籌莫展，比如希臘字母 (Greek) 和西里爾字母 (Cyrillic)。

爲了克服這個缺點, 一些 8-bit 的類似 CM 的字體集被打造出來。 `T1` 編碼的 *Extended Cork* (EC) 字體以拉丁語係爲基礎，包含了支持大部分歐洲語言

[5] 要想知道更多基於 Latin 或者 Cyrillic 語言支持的輸入編碼，請分別閱讀 `inputenc.dtx` 和 `cyinpenc.dtx` 的文檔。第 4.6 節講到了如何生成宏包文檔。

表 2.3 – 葡萄牙文所需的導言區。

```
\usepackage[portuguese]{babel}
\usepackage[latin1]{inputenc}
\usepackage[T1]{fontenc}
```

的字母和標點符號。 LH 字體集包含了排版斯拉夫語系文檔必需的字母。因爲斯拉夫字母的字形太多，它們被分成四種字體編碼 —— T2A, T2B, T2C, 以及 X2[6]。希臘文的 LGR 編碼字體在 CB 字體集裡。

有了這些字體支持，你可以對非英文文本改進或者應用斷詞了。使用這些新的類似 CM 的字體還有一個好處，它們提供了 CM 字族裡各種大小，形狀以及比例縮放的字體。

2.5.1 葡萄牙文支持

By Demerson Andre Polli <polli@linux.ime.usp.br>

爲了對葡萄牙文 (Portuguese) 文檔應用斷詞及各種自動文本，使用命令：

```
\usepackage[portuguese]{babel}
```

或者如果你在巴西的話，替換成 brazilian。

鑑於葡萄牙文中有許多重音，你可能想要用

```
\usepackage[latin1]{inputenc}
```

來正確的輸入它們，並且用

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

來正確的斷詞.

使用葡萄牙文的文檔導言區請參考表 2.3。注意我們使用的是 latin1 的輸入編碼，所以在 Mac 或者 DOS 上會不起作用。請自行選擇合適的編碼。

2.5.2 法文支持

By Daniel Flipo <daniel.flipo@univ-lille1.fr>

一些使用 LaTeX 創建法文 (French) 文檔的提示：你可以通過以下命令載入法文支持：

```
\usepackage[frenchb]{babel}
```

請注意，由於歷史原因，babel 的法文選項或者是 *frenchb* 或者是 *francais*，而不是 *french*。

照此設定，你就可以使用法文的斷詞了。當然所有的自動文本也都成爲法文： \chapter 印成 Chapitre, \today 印成法語裡的今天的日期等等。同時也有一系列的新命令，可以讓你更容易的輸入法文。請參考表 2.4 來獲取靈感。

你會注意到，切換到法文的時候，列表的版面也改變了。更多關於 babel 的 frenchb 選項功能以及如何定製的內容，請對 frenchb.dtx 運行 LaTeX 並閱讀生成的 frenchb.dvi。

[6]這些編碼所支持的語言列表可以在 [11] 查到。

表 2.4 – 法文專用命令。

`\og guillemets \fg{}`	« guillemets »
`M\up{me}, D\up{r}`	M^{me}, D^{r}
`1\ier{}, 1\iere{}, 1\ieres{}`	1er, 1re, 1res
`2\ieme{} 4\iemes{}`	2^{e} 4es
`\No 1, \no 2`	N^{o} 1, n^{o} 2
`20 \degres C, 45\degres`	20 °C, 45°
`\bsc{M. Durand}`	M. Durand
`\nombre{1234,56789}`	1 234,567 89

2.5.3 德文支持

一些使用 LaTeX 創建德文 (German) 文檔的提示：你可以通過以下命令來載入德文支持：

```
\usepackage[german]{babel}
```

照此設定，你就可以使用德文的斷詞了。當然所有的自動文本也都成爲德文：例如 “Chapter” 印成 “Kapitel”。同時也有一系列的新命令，可以讓你更迅速的輸入德文，即使你沒有使用 inputenc 宏包。請參考表 2.5 來獲取靈感。一旦使用 inputenc 宏包，所有這些都不重要了，當然你的文檔也被鎖定在一個特殊的編碼世界裡。

表 2.5 – 德文專用字符。

`"a`	ä	`"s`	ß
`"‘`	„	`"’`	“
`"< or \flqq`	«	`"> or \frqq`	»
`\flq`	‹	`\frq`	›
`\dq`	”		

在德文的書籍裡，你會經常發現法文的引號 («guillemets»)。然而德文的打字機裡有不同的使用方法。德文書籍中的引號看起來是 »this«。在瑞士講德語的部分，打字機使用 «guillemets»，這跟法文一樣。

使用類似 `\flq` 命令的一個主要問題是：如果你用 OT1 字體 (這是默認字體)， guillemets 看起來就像數學符號 “≪”，這令排版者反胃。而 T1 編碼的字體含有正確的符號。所以，當你使用這種引號的時候，請確保正在用 T1 編碼。 (`\usepackage[T1]{fontenc}`)

2.5.4 朝鮮文支持[7]

爲了使用 LaTeX 排版韓文 (Korean)，我們需要解決三個問題：

1. 我們要能夠編輯韓文的源文件 (Korean input files)。韓文源文檔必須是普通文本格式的 (plain-text format)，但由於韓文使用的字符集迥異於 US-ASCII 指令集，在一般的 ASCII 編輯器裡看起來會相當怪異。兩個最廣爲使用的韓文文本文檔編碼是 EUC-KR 以及 MS-Windows 裡它的向上兼容擴展，CP949/Windows-949/UHC。在這些編碼裡，每一個 US-ASCII 字符代表普通的 ASCII 字符，這跟其他兼容 ASCII 的編碼比如 ISO-8859-x，EUC-JP， Big5，或者 Shift_JIS 相似。另一方面，從 KS X 1001 字符編碼取出的韓語諺文、漢字、朝鮮文字母、平假名、片假名、希臘文和斯拉夫字符以及其他符號和字母都用兩個連貫的八位字節來表示。第一種有它的有效位集。直到 1990 年代中期，在非韓文的操作系統上設定朝鮮文兼容環境還是一件費時費力的事。你可以瀏覽一下有些過時的 http://jshin.net/faq 來瞭解那時是如何在非韓文操作系統上使用朝鮮文的。現在，三種主要的操作系統 (Mac OS, Unix, Windows) 都具備了相當好的多語言支持和國際化特徵，所以在非韓文平台上編輯朝鮮文文檔已經不再是一個問題了。

2. TeX 和 LaTeX 最初只支持不超過 256 個字符。爲了在其他有大量字符的語文例如韓文或漢文中讓它們工作 [8] ，開發了一種子字體機制。一個有幾千或者幾萬種字型 (glyph) 的 CJK 單字被分割成一組子字體集，每一集合裡包含 256 個字型。對韓文而言，有三個廣爲使用的宏包：UN Koaunghi 開發的 HLaTeX， CHA Jaechoon 的 hLaTeXp 以及 Werner Lemberg 的 CJK 宏包 (CJK package)[9]。 HLaTeX 和 hLaTeXp 專爲韓文設計並且在字體支持之外支持朝鮮文本地化 (Korean localization)。對於 EUC-KR 編碼的源文檔，它們都可以正確的處理。在使用 Λ 和 Ω 的時候， HLaTeX 還可以處理以 CP949/Windows-949/UHC 和 UTF-8 編碼的源文檔。

   CJK 宏包不只爲韓文提供支持。它還可以處理以 UTF-8 以及很多 CJK 編碼包括 EUC-KR 和 CP949/Windows-949/UHC 的源文檔。它支持多種語言內容的文檔排版，特別是漢文，日文和韓文。跟 HLaTeX 相比，CJK 宏包不提供韓文本地化而且朝鮮文字體也不如 HLaTeX 多。

3. 使用如 TeX 和 LaTeX 排版工具的最終目的是用「美學」上令人滿意的方式排版文檔。可以說，排版中最重要的是優美設計的字體。 HLaTeX 發

[7]考慮到韓文 LaTeX 用戶需要處理的大量問題， Karnes KIM 代表韓國 lshort 翻譯團隊撰寫了這一節，並由 SHIN Jungshik 翻譯爲英文， Tobi Oetiker 作了簡化。

[8]韓語諺文是一種由 14 個基本輔音和 10 個基本元音構成的字母書寫系統。不同於拉丁或者斯拉夫文字，每一個字符都要被排進跟漢文字符差不多大小的一簇矩形裡。每一簇表示一個音節。這樣就用有限的元音和輔音構成了無限多的音節。但是現代韓文的拼寫標準（南、北朝鮮）都對這些簇的構成有嚴格的限制。因此只有有限個拼寫正確的音節存在。韓文字符編碼給每一個音節的指定一個代碼 (KS X 1001:1998 和 KS X 1002:1992)。所以諺文雖然是一種字母文，處理起來卻跟漢文和日文這些有幾萬個表意字符的書寫系統差不多。 ISO 10646/Unicode 提供了現代韓語諺文的兩種表示方法，一種是對相連的諺文字母編碼（字母表： http://www.unicode.org/charts/PDF/U1100.pdf ），另一種對所有拼寫規範的現代韓語音節編碼 (http://www.unicode.org/charts/PDF/UAC00.pdf)。使用 LaTeX 及其相關排版系統處理韓文有一項最令人犯　的挑戰，就是對中古朝鮮文 —— 可能會成爲未來的韓文 —— 音節的支持，現在還只能用 Unicode 對相連的字母編碼來解決。希望未來的 TeX 引擎如 Ω 和 Λ 會最終提供解決方案，使得韓語言和歷史學者丟開 MS Word，雖然它已經對中古朝鮮文有了良好的支持。

[9]這些可以在 `language/korean/HLaTeX/`, `language/korean/CJK/` 和 `http://knot.kaist.ac.kr/htex/` 取得。

行版包含 10 族 (family) UHC PostScript 字體和 5 族 (family) 文化部 (Munhwabu[10]) 字體 (TrueType)。 CJK 宏包使用的字體是 HLaTeX 較早版本裡的，但它支持 Bitstream's cyberbit TrueType 字體。

使用 HLaTeX 宏包來輸入韓文，只需把

```
\usepackage{hangul}
```

放到你的導言區即可。

這一命令啓動了韓文本地化支持。章、節、子節、目錄和圖表目錄都會被轉換成相應的朝鮮文，而且使用韓文的習慣來格式化文檔。這個宏包還提供了自動的「 虛詞選擇 」功能。在韓文裡，有大量的這類語法上等價但是形式不同的後綴虛詞，哪一個詞組組合是正確的依賴於前面的音節是以元音還是以輔音結尾的。（ 實際情況比這還要複雜，但上述描述足夠給你一個大致的印象了 ）以韓文爲母語的人選擇適合的虛詞毫無問題，但是文檔編輯中隨時改變的參考文獻以及其他自動文本就很難確定。每一次你增刪參考文獻或者改變文檔內容的順序時，手工放置合適的虛詞都是一件辛苦的工作。 HLaTeX 的用戶就可以從這種煩人而且容易出錯的工作中解放出來。

如果你不需要韓文本地化，只是想要排版一些朝鮮文字，可以把放到導言區的命令換成：

```
\usepackage{hfont}
```

更多使用 HLaTeX 排版韓文的信息，請看 *HLaTeX Guide*。訪問 Korean TeX User Group (KTUG) 的網頁 http://www.ktug.or.kr/。那裡也有一份本手册的韓語譯本。

2.5.5 用希臘文寫作

By Nikolaos Pothitos <pothitos@di.uoa.gr>

使用希臘文 (Greek) 寫作所需的導言內容參見表 2.6。它們可以實現希臘文的斷詞和自動文本[11]。

表 2.6 – 希臘文文檔所需導言區。

```
\usepackage[english,greek]{babel}
\usepackage[iso-8859-7]{inputenc}
```

有一組新的命令可以讓你更容易地輸入希臘文。爲了暫時切換爲英文或者相反，你可以使用命令 `\textlatin{`*english text*`}` 以及 `\textgreek{`*greek text*`}`，它們都只有一個參量，可以使用所要求的字體編碼排版。或者你也可以使用前面章節說過的命令 `\selectlanguage{...}`。表 2.7 列出了一些希臘文標點符號。對於歐元符號，要使用 `\euro`。

[10] 南韓文化部。

[11] 如果對 `inputenc` 宏包使用了 `utf8x` 選項，你可以排版希臘文和多聲調希臘文的 unicode 字符。

表 2.7 – 希臘文特殊字符。

;	·	?	;
((	«	))	»
‘‘	‘	,,	’

2.5.6 斯拉夫文支持

By Maksym Polyakov <polyama@myrealbox.com>

版本爲 3.7h 的 babel 宏包包含了對 T2* 編碼以及使用斯拉夫字母排版保加利亞文、俄文和烏克蘭文的支持。

斯拉夫文的支持依賴於 LaTeX 系統還有 fontenc 和 inputenc 宏包。但是如果你要在數學模式下使用斯拉夫文，就必須在 inputenc 之前加載 mathtext 宏包 [12]:

```
\usepackage{mathtext}
\usepackage[T1,T2A]{fontenc}
\usepackage[koi8-ru]{inputenc}
\usepackage[english,bulgarian,russian,ukranian]{babel}
```

一般情況下，babel 會自動選擇的默認的字體編碼，對於上面三種語文，應該是 T2A。然而，文檔不會限制只使用一種字體編碼。對於有拉丁語系和斯拉夫語系的多語文文檔，應該明確包含拉丁語文字體的編碼。在文檔中，當選擇另外一種語文的時候，babel 會控制切換到合適的字體編碼。

除了能夠斷詞, 翻譯自動文本字符串，以及啓動一些語文專用的排版規則（比如 \frenchspacing）， babel 還提供了一些命令可以按照保加利亞文、俄文、或者烏克蘭文的標準排版。

這三種語言專用的標點符號也被提供了：斯拉夫文本的破折號（它比拉丁語文的破折號略窄，周圍有微小的空白）、直接引語用的破折號、引號、以及方便斷詞的命令，請參考表 2.8。

babel 的 Russian 和 Ukrainian 選項定義了命令 \Asbuk 和 \asbuk，它們的作用類似於 \Alph 和 \alph，產生俄文和烏克蘭文的大寫和小寫字母（無論文檔的活動語言是哪一個）。babel 的 Bulgarian 選項提供了命令 \enumBul 和 \enumLat (\enumEng)，它們可以讓 \Alph 和 \alph 產生保加利亞文或者拉丁（英文）字母的大小寫，默認爲保加利亞文的。

2.6 單詞間隔

爲了使輸出的右邊界對齊，LaTeX 在單詞間插入不等的間隔。在句子的末尾插入的空間稍多一些，因爲這使得文本更具可讀性。 LaTeX 假定句子以句號、問號或驚嘆號結尾。如果句號緊跟一個大寫字母，它就不視爲句子的結尾。因爲一般在有縮寫的地方，才出現句號緊跟大寫字母的情況。

作者必須詳細說明這些假設中的任何一個例外。空格前的反斜線符號產生一個不能伸長的空格。波浪字符‘ ’也產生一個不能伸長的空格，並且禁止

[12]如果使用了 $\mathcal{AMS}$-LaTeX 的宏包，相應的把它們放在 fontenc 和 babel 之前加載。

表 2.8 – babel 的 Bulgarian、 Russian 和 Ukrainian 選項一些額外的定義。

"|	當前位置取消連字。
"-	一個明確的斷詞符號，允許在單詞的其他位置斷詞。
"---	普通斯拉夫文本中的破折號。
"--	合成的姓名（姓）中用的破折號。
"--*	表示直接引語的斯拉夫文破折號。
""	類似於 "-，但是不產生連字號（用於合成詞中，比如 x-""y 或者或者其他像 "enable/disable" 的符號）。
"	沒有斷開點的合成詞標記。
"=	帶斷開點的合成詞標記，允許在構成單詞裡斷詞。
",	短的空白，用於帶斷開點的姓的首字母。
"‘	用於德文裡的左雙引號（看起來像„）。
"’	用於德文裡的右雙引號（看起來像 “）。
"<	用於法文的左雙引號（看起來像 ≪）。
">	用於法文的右雙引號（看起來像 ≫）。

斷行。句號前的命令 \@ 說明這個句號是句子的末尾，即使它緊跟一個大寫字母。

```
Mr. Smith was happy to see her\\
cf. Fig. 5\\
I like BASIC\@. What about you?
```

Mr. Smith was happy to see her
cf. Fig. 5
I like BASIC. What about you?

命令

```
\frenchspacing
```

能禁止在句號後插入額外的空白，它告訴 LaTeX 在句號後不要插入比正常字母更多的空白。除了參考文獻，這在非英語語言中非常普遍。如果使用了 \frenchspacing，命令 \@ 就不必要了。

2.7 標題、章和節

爲便於讀者理解，應該把文檔劃分爲章，節和子節。LaTeX 用專門的命令支持這個工作，這些命令把節的標題作爲參量。你的任務是按正確次序使用它

們。對 article 風格的文檔，有下列分節命令：

```
\section{...}
\subsection{...}
\subsubsection{...}
\paragraph{...}
\subparagraph{...}
```

如果想把文檔分成幾個部分而且不影響章節編號，你可以使用

```
\part{...}
```

當你使用 report 或者 book 類的時候，可以用另外一個高層次的分節命令

```
\chapter{...}
```

因爲 article 類的文檔不劃分爲章，所以很容易把它作爲一章插入書籍中。節之間的間隔，節的序號和標題的字號由 LaTeX 自動設置。

分節的兩個命令有些特別：

- 命令 \part 不影響章的序號。
- 命令 \appendix 不帶參量，只把章的序號改用爲字母標記 [13]。

LaTeX 在文檔編譯的最後一個循環中，提取節的標題和頁碼以生成目錄。命令

```
\tableofcontents
```

在其出現的位置插入目錄。爲了得到正確的目錄 (table of contents) 內容，一個新文檔必須編譯 ("LaTeXed") 兩次。有時還要編譯第三次。如有必要 LaTeX 會告訴你。

上面列出的分節命令也以「帶星」的形式出現。「帶星」的命令通過在命令名稱後加 * 來實現。它們生成的節標題既不出現於目錄，也不帶序號。例如，命令 \section{Help} 的「帶星」形式爲 \section*{Help}。

目錄出現的標題，一般與輸入的文本完全一致。有時這是不可能的，因爲標題太長排不進目錄。在這種情況下，目錄的條目可由實際標題前的可選參量確定。

```
\chapter[Title for the table of contents]{A long
    and especially boring title, shown in the text}
```

整篇文檔的標題 (title) 由命令

```
\maketitle
```

產生。標題的內容必須在調用 \maketitle 以前，由命令

```
\title{...}, \author{...} 和可選的 \date{...}
```

定義。在命令 \author 的參量中，可以輸入幾個用 \and 命令分開的名字。

在第 6 頁的圖 1.2 中，能找到有關上述命令的一個例子。

除了上面解釋的分節命令，LaTeX 2_ε 引進了其他三個命令用於 book 風格的文檔。它們對劃分出版物有用，也能如願改變章的標題和頁碼：

[13] 對 article 類文檔改變節的序號。

\frontmatter 應接著命令 \begin{document} 使用。它把頁碼更換爲羅馬數字，而且章節不計數。當你使用帶星的分節命令 (例如，\chapter*{Preface}) 時，這些章節就不會出現在目錄裡。

\mainmatter 應出現在書的第一章前面。它啓用阿拉伯數字的頁碼計數器，並對頁碼重新計數。

\appendix 標誌書中附錄材料的開始。該命令後的各章序號改用字母標記。

\backmatter 應該插入與書中最後一部分內容的前面，如參考文獻和索引。在標準文檔類型中，它對頁面沒有什麼效果。

2.8 交叉引用

在書籍、報告和論文中，需要對圖、表和文本的特殊段落進行交叉引用 (cross-references)。 LaTeX 提供了如下交叉引用命令

\label{*marker*}, \ref{*marker*} 和 \pageref{*marker*}

其中 *marker* 是用戶選擇的標識符。如果在節、子節、圖、表或定理後面輸入 \label 命令，LaTeX 把 \ref 替換爲相應的序號。 \pageref 命令排印 \label 輸入處的頁碼[14]。和章節標題一樣，使用的序號是前面編譯所產生。

```
A reference to this subsection
\label{sec:this} looks like:
``see section \ref{sec:this} on
page \pageref{sec:this}.''
```

A reference to this subsection looks like: “see section 2.8 on page 28.”

2.9 腳註

命令

\footnote{*footnote text*}

把脚註內容排印於當前頁的頁脚位置。脚註命令總是置於 (put) [15] 其指向的單詞或句子的後面。脚註是一個句子或句子的一部分，所以應用逗號或句號結尾[16]。

```
Footnotes\footnote{This is
  a footnote.} are often used
by people using \LaTeX.
```

Footnotes[a] are often used by people using LaTeX.

[a] This is a footnote.

[14] 注意這些命令對它們指向什麼並沒有意識。命令 \label 只是保存了上一次自動產生的序號。

[15] “put” 是最常使用的英文單詞之一。

[16] 注意，脚註把讀者的注意力從文檔的正文引開。我們是好奇的動物，每個人都會閱讀脚註。所以爲什麼不把你想說的所有東西都寫入正文中？[17]

[17] 路標不必走向它指向的地方 :-)。

2.10 強調

如果文本是用打字機鍵入的， 用下劃線來強調重要的單詞。

`\underline{`*text*`}`

但是在印刷的書中，用一種斜體字體排印要強調的單詞。 LaTeX 提供命令

`\emph{`*text*`}`

來強調文本。這些命令對其參量的實際作用效果依賴於它的上下文：

```
\emph{If you use
  emphasizing inside a piece
  of emphasized text, then
  \LaTeX{} uses the
  \emph{normal} font for
  emphasizing.}
```

If you use emphasizing inside a piece of emphasized text, then LaTeX uses the normal *font for emphasizing.*

請注意要求 LaTeX **強**調什麼和要求它使用不同**字**體的不同效果:

```
\textit{You can also
  \emph{emphasize} text if
  it is set in italics,}
\textsf{in a
  \emph{sans-serif} font,}
\texttt{or in
  \emph{typewriter} style.}
```

You can also emphasize *text if it is set in italics,* in a *sans-serif* font, or in *typewriter* style.

2.11 環境

爲了排版專用的文本，LaTeX 定義了各種不同格式的環境 (environment)：

`\begin{`*environment*`}` *text* `\end{`*environment*`}`

其中 *environment* 是環境的名稱。只要保持調用順序，環境可以嵌套。

```
\begin{aaa}...\begin{bbb}...\end{bbb}...\end{aaa}
```

下面的章節對所有重要的環境都做瞭解釋。

2.11.1 Itemize、Enumerate 和 Description

`itemize` 環境適用於簡單的列表，`enumerate` 環境適用於有排列序號的列表，而 `description` 環境用於帶描述的列表。

```
\flushleft
\begin{enumerate}
\item You can mix the list
environments to your taste:
\begin{itemize}
\item But it might start to
look silly.
\item[-] With a dash.
\end{itemize}
\item Therefore remember:
\begin{description}
\item[Stupid] things will not
become smart because they are
in a list.
\item[Smart] things, though,
can be presented beautifully
in a list.
\end{description}
\end{enumerate}
```

1. You can mix the list environments to your taste:
   - But it might start to look silly.
   - With a dash.
2. Therefore remember:

   Stupid things will not become smart because they are in a list.

   Smart things, though, can be presented beautifully in a list.

2.11.2 左對齊、右對齊和居中

`flushleft` 和 `flushright` 環境分別產生左對齊 (left-aligned) 和右對齊 (right-aligned) 的段落。`center` 環境產生居中的文本。如果你不輸入命令 `\\` 指定斷行點， LaTeX 將自行決定。

```
\begin{flushleft}
This text is\\ left-aligned.
\LaTeX{} is not trying to make
each line the same length.
\end{flushleft}
```

This text is
left-aligned. LaTeX is not trying to make each line the same length.

```
\begin{flushright}
This text is right-\\aligned.
\LaTeX{} is not trying to make
each line the same length.
\end{flushright}
```

This text is right-
aligned. LaTeX is not trying to make each line the same length.

```
\begin{center}
At the centre\\of the earth
\end{center}
```

At the centre
of the earth

2.11.3 引用、語錄和韻文

`quote` 環境可以用於引文、語錄和例子。

```
A typographical rule of thumb
for the line length is:
\begin{quote}
On average, no line should
be longer than 66 characters.
\end{quote}
This is why \LaTeX{} pages have
such large borders by default
and also why multicolumn print
is used in newspapers.
```

A typographical rule of thumb for the line length is:

> On average, no line should be longer than 66 characters.

This is why LaTeX pages have such large borders by default and also why multicolumn print is used in newspapers.

有兩個類似的環境：quotation 和 verse 環境。quotation 環境用於超過幾段的較長引用，因爲它對段落進行縮進。verse 環境用於詩歌，在詩歌中斷行很重要。在一行的末尾用 \\ 斷行，在每一段後留一空行。

```
I know only one English poem by
heart. It is about Humpty Dumpty.
\begin{flushleft}
\begin{verse}
Humpty Dumpty sat on a wall:\\
Humpty Dumpty had a great fall.\\
All the King's horses and all
the King's men\\
Couldn't put Humpty together
again.
\end{verse}
\end{flushleft}
```

I know only one English poem by heart. It is about Humpty Dumpty.

Humpty Dumpty sat on a wall:
Humpty Dumpty had a great fall.
All the King's horses and all the King's men
Couldn't put Humpty together again.

2.11.4 摘要

科學出版物慣常以摘要開始，來給讀者一個綜述或者預期。 LaTeX 爲此提供了 abstract 環境。一般 abstract 用於 article 類文檔。

```
\begin{abstract}
The abstract abstract.
\end{abstract}
```

The abstract abstract.

2.11.5 原文照列

位於 \begin{verbatim} 和 \end{verbatim} 之間的文本將直接列印，包括所有的斷行和空白，就像在打字機上鍵入一樣，不執行任何 LaTeX 命令。

在一個段落中，類似的功能可由

\verb+*text*+

完成。+ 僅是分隔符的一個例子。除了 * 或空格，可以使用任意一個字符。這個小册子中的許多例子是用這個命令排印的。

```
The \verb|\ldots| command \ldots
\begin{verbatim}
10 PRINT "HELLO WORLD ";
20 GOTO 10
\end{verbatim}
```

The \ldots command ...

```
10 PRINT "HELLO WORLD ";
20 GOTO 10
```

```
\begin{verbatim*}
the starred version of
the      verbatim
environment emphasizes
the spaces   in the text
\end{verbatim*}
```

```
the starred version of
the      verbatim
environment emphasizes
the spaces   in the text
```

帶星的命令 \verb 能以類似的方式使用：

```
\verb*|like   this :-) |
```

```
like   this :-)
```

verbatim 環境和 \verb 命令不能在其他命令的參數中使用。

2.11.6 表格

tabular 環境能用來排版帶有水平和垂直表線的漂亮表格 (table)。LaTeX 自動確定每一列的寬度。

命令

\begin{tabular}[*pos*]{*table spec*}

的參量 *table spec* 定義了表格的格式。用一個 l 產生左對齊的列，用一個 r 產生右對齊的列，用一個 c 產生居中的列；用 p{*width*} 產生相應寬度、包含自動斷行文本的列； | 產生垂直表線。

如果一列裡的文本太寬， LaTeX 不會自動折行顯示。使用 p{*width*} 你可以定義如一般段落裡折行效果的列。

參量 *pos* 設定相對於環繞文本基線的垂直位置。使用字母 t、 b 和 c 來設定表格靠上、靠下或者居中放置。

在 tabular 環境中，用 & 跳入下一列，用 \\ 開始新的一行，用 \hline 插入水平表線。用 \cline{*j*-*i*} 可添加部分表線，其中 j 和 i 分別表示表線的起始列和終止列的序號。

```
\begin{tabular}{|r|l|}
\hline
7C0 & hexadecimal \\
3700 & octal \\ \cline{2-2}
11111000000 & binary \\
\hline \hline
1984 & decimal \\
\hline
\end{tabular}
```

7C0	hexadecimal
3700	octal
11111000000	binary
1984	decimal

```
\begin{tabular}{|p{4.7cm}|}
\hline
Welcome to Boxy's paragraph.
We sincerely hope you'll
all enjoy the show.\\
\hline
\end{tabular}
```

Welcome to Boxy's paragraph. We sincerely hope you'll all enjoy the show.

表格的列分隔符可由 `@{...}` 構造。這個命令去掉表列之間的間隔，代之爲兩個花括號間的內容。一個用途在於下面要解釋的十進制數對齊問題。另一個可能應用在於用 `@{}` 壓縮表列右端空間。

```
\begin{tabular}{@{} l @{}}
\hline
no leading space\\
\hline
\end{tabular}
```

no leading space

```
\begin{tabular}{l}
\hline
leading space left and right\\
\hline
\end{tabular}
```

leading space left and right

由於沒有內建機制使十進制數按小數點對齊[18]，我們可以使用兩列「作弊」達到這個目的：整數向右，小數向左對齊。`\begin{tabular}` 行中的命令 `@{.}` 用一個 "." 取代了列間正常間隔，從而給出了按小數點列對齊的效果。不要忘記用列分隔符 (`&`) 取代十進制小數點！使用命令 `\multicolumn` 可在數值「列」上放置一個列標籤。

```
\begin{tabular}{c r @{.} l}
Pi expression       &
\multicolumn{2}{c}{Value} \\
\hline
$\pi$               & 3&1416  \\
$\pi^{\pi}$         & 36&46   \\
$(\pi^{\pi})^{\pi}$ & 80662&7 \\
\end{tabular}
```

Pi expression	Value
π	3.1416
π^{π}	36.46
$(\pi^{\pi})^{\pi}$	80662.7

```
\begin{tabular}{|c|c|}
\hline
\multicolumn{2}{|c|}{Ene} \\
\hline
Mene & Muh! \\
\hline
\end{tabular}
```

Ene	
Mene	Muh!

用表格環境排印的材料總是呆在同一頁上。如果要排印一個長表格，可以看一下 supertabular 和 longtabular 環境。

[18]如果係統安裝了 'tools' 包，請看一下宏包 dcolumn。

2.12 浮動體

今天大多數出版物含有許多圖片和表格。由於不能把它們分割在不同的頁面上，所以需要專門的處理。如果一個圖片或一個表格太大在當前頁面排不下，一個解決辦法就是每次新開一頁。這個方法在頁面上留下部分空白，效果看起來很差。

對於在當前排不下的任何一個圖片或表格，其解決辦法是把它們「浮動」到下一頁，與此同時當前頁面用正文文本填充。LaTeX 提供了兩個浮動體 (floating bodies) 環境；一個用於圖片，一個用於表格。要充分發揮這兩個環境的優越性，應該大致瞭解 LaTeX 處理浮動體的內在原理。但是浮動可能成爲令人沮喪的主要原因，因爲 LaTeX 總不把浮動體放在你想要的位置。

首先看一下供浮動使用的 LaTeX 命令：

包含在 `figure` 環境或 `table` 環境中的任何材料都將被視爲浮動內容。兩個浮動環境都支持可選參數

`\begin{figure}[`*placement specifier*`]` 或 `\begin{table}[...]`

稱爲 *placement specifier*，它由**浮**動許可放置參數寫成的字符串組成。請見表 2.9。這個參數用於告訴 LaTeX 浮動體可以被移放的位置。一個 *placement specifier* 由一串**浮**動體許可放置位置 (*float-placing permissions*) 構成. 參見表 2.9。

一個表格可以由如下命令，例如

```
\begin{table}[!hbp]
```

開始，placement specifier `[!hbp]` 允許 LaTeX 把表格就放當前頁，或放在某頁的底部 (`b`)，或放在一個專門的浮動頁上 (`p`)，嚴格按照放置說明符放置即使看起來不好 (`!`)。如果沒有給定放置說明符，缺省值爲 `[tbp]`。

LaTeX 將按照作者提供的 placement specifier ，安排它遇到的每一個浮動體。如果浮動體在當前頁不能安排，就把它寄存在**圖**片或**表**格等待隊列中[19]。當新的一頁開始的時候，LaTeX 首先檢查是否可能用等待隊列中的浮動體填充一個專門的「浮動」頁面。如果這不可能，就像對待剛在文本中出現的浮動體一樣，處理等待隊列中的第一個浮動體：LaTeX 重新嘗試按照其相應的放置說

[19] 它們是「先來先走」隊列！

表 2.9 – 浮動體放置許可。

Spec	浮動體許可放置位置… …
`h`	*here* 在文本的確切位置上，對於小的浮動體很有用。
`t`	在頁面的頂部 (*top*)
`b`	在頁面的底部 (*bottom*)
`p`	在一個只有浮動體的專門的頁面 (*page*) 上。
`!`	忽略阻止浮動體放置的大多數內部參數[a]。

注意 `pt` 和 `em` 是 TeX 單位。請閱讀第 92 頁上表 6.5 更多有關的更多內容。

[a] 例如一頁上所允許的浮動體的最大數目。

明符 (除了不再可能的 'h') 來處理它。文本中出現的任何一個新浮動體寄存在相應的等待隊列中。對於每一種浮動體，LaTeX 保持它們出現的順序。這就說明了爲什麼一個不能安排的圖片把所有後來的圖片都推到文檔末尾的原因。所以：

> 如果 LaTeX 沒有像你期望的那樣安排浮動體，那麼經常是僅有一個浮動體堵塞了兩個等待隊列中的某一個。

僅給定單個 placement specifiers 是允許的，但這會引起問題。如果在指定的位置安排不了，它就會成爲障礙，堵住後續的浮動體。不要單獨使用參數 [h]，在 LaTeX 最近的版本中，它的效果太差了以至於被 [ht] 自動替換。

雖然對浮動體問題已經作了些說明，對 `table` 和 `figure` 環境還有些內容要交代。使用

`\caption{`*caption text*`}`

命令，可以給浮動體定義一個標題。序號和字符串「 圖 」或「 表 」將由 LaTeX 自動添加。

兩個命令

`\listoffigures` 和 `\listoftables`

用起來和 `\tableofcontents` 命令類似，分別排版一個圖形目錄和表格目錄。在這些目錄中，所有的標題都將重複。如果打算使用長標題，就必須準備一個能放進目錄的，較短版本的標題。即在 `\caption` 命令後面的括號內輸入較短版本的標題。

```
\caption[Short]{LLLLLooooooonnnnnggggg}
```

利用 `\label` 和 `\ref`，在文本中可以爲浮動體創建交叉引用。

下面的例子畫一個方形，並將它插入文檔。如果想在完成的文檔中爲你打算嵌入的圖片保留空間，你可以利用這個例子。

```
Figure \ref{white} is an example of Pop-Art.
\begin{figure}[!hbp]
\makebox[\textwidth]{\framebox[5cm]{\rule{0pt}{5cm}}}
\caption{Five by Five in Centimetres.\label{white}}
\end{figure}
```

在上面的例子中，爲了把圖片**就**放在當前位置 (h)[20]， LaTeX 嘗試得**很**辛苦 (!)。如果這不可能，它將試圖把圖片安排在頁面的**底**部 (b)。如果不能將圖片安排在當前頁面，它將決定是否可能開一個浮動頁面以放置這張圖片或來自表格等待隊列中的一些表格。如果沒有足夠的材料來填充一個專門浮動頁面，LaTeX 就開一個新頁，像對文本中剛出現的圖片一樣，再一次處理這個圖片。

在一些情況下，可能需要使用命令

`\clearpage` 或者甚至是 `\cleardoublepage`

它命令 LaTeX 立即放置等待隊列中所有剩下的浮動體，並且開一新頁。命令 `\cleardoublepage` 甚至會命令 LaTeX 新開奇數頁面。

在本書的後面，將介紹如何在 LaTeX 2$_\varepsilon$ 文檔中插入 PostScript 圖形。

[20] 假設圖片的等待隊列已空。

2.13 保護脆弱命令

作爲命令（如 \caption 或 \section）參量的文本，可能在文檔中出現多次（例如，在文檔的目錄和正文中）。當用於類似 \section 的參量時，一些命令會失效。它們被稱爲脆弱命令 (fragile commands)。\footnote 或 \phantom 是脆弱命令的例子。這些脆弱命令需要的，正是保護。把 \protect 命令放在它們前面，就能保護它們。

\protect 僅僅保護緊跟其右側的命令，連它的參量也不惠及。在大多數情形下，過多的 \protect 並不礙事。

```
\section{I am considerate
      \protect\footnote{and protect my footnotes}}
```

Chapter 3

數學公式

現在你已經準備好了。那麼在這一章裡，讓我們來著手於 TEX 的強大之處：數學排版。但是，要提醒你的是，本章只是淺嘗輒止。可對很多人來說，這裡所講述的內容已很受用，如果你在這裡找不到你所需數學排版的解決方案的話，也請不要灰心。極有可能在 AMS-LATEX[1] 中能找到針對你的問題的某個解決方案。

3.1 綜述

LATEX 使用一種特有的模式來排版數學 (mathematics) 公式。數學公式允許以行間形式排版在一個段落之中，也可以以獨立形式排版，此時段落可能會被拆開。處於段內的數學文本要放在 \(與 \) 之間， $ 與 $ 之間，或者 \begin{math} 與 \end{math} 之間。

```
Add $a$ squared and $b$ squared
to get $c$ squared. Or, using
a more mathematical approach:
$c^{2}=a^{2}+b^{2}$
```

Add a squared and b squared to get c squared. Or, using a more mathematical approach: $c^2 = a^2 + b^2$

```
\TeX{} is pronounced as
\(\tau\epsilon\chi\).\\[6pt]
100 m$^{3}$ of water\\[6pt]
This comes from my
\begin{math}\heartsuit\end{math}
```

TEX is pronounced as $\tau\epsilon\chi$.

100 m^3 of water

This comes from my $\heartsuit$

當你希望把自己的一些較長的數學方程或是公式單獨的放在段落之外的時候，那麼你最好顯示 (display) 它們，而不要拆開此段落。爲此，你可以把它們放在 \[與 \] 之間，或者 \begin{displaymath} 與 \end{displaymath} 之間。

[1] 美國數學學會 製作了一個強大的 LATEX 擴展。本章的很多例子都使用了這個擴展。所有最近的 TEX 發行版中都提供了這個擴展。如果你的系統中沒有，可以去 macros/latex/required/amslatex 找找看。

```
Add $a$ squared and $b$ squared
to get $c$ squared. Or, using
a more mathematical approach:
\begin{displaymath}
c^{2}=a^{2}+b^{2}
\end{displaymath}
or you can type less with:
\[a+b=c\]
```

Add a squared and b squared to get c squared. Or, using a more mathematical approach:

$$c^2 = a^2 + b^2$$

or you can type less with:

$$a + b = c$$

如果你希望 LaTeX 給你的方程編上號，你可以使用 `equation` 環境。然後你就可以用 `\label` 來給一個方程加上標籤並在文中的某處用 `\ref` 或 amsmath 宏包中的 `\eqref` 命令來引用它。

```
\begin{equation} \label{eq:eps}
\epsilon > 0
\end{equation}
From (\ref{eq:eps}), we gather
\ldots{}From \eqref{eq:eps} we
do the same.
```

$$\epsilon > 0 \tag{3.1}$$

From (3.1), we gather ... From (3.1) we do the same.

注意一下公式排版樣式的不同，前者是行間式樣，後者是顯示式樣：

```
$\lim_{n \to \infty}
\sum_{k=1}^n \frac{1}{k^2}
= \frac{\pi^2}{6}$
```

$\lim_{n\to\infty} \sum_{k=1}^{n} \frac{1}{k^2} = \frac{\pi^2}{6}$

```
\begin{displaymath}
\lim_{n \to \infty}
\sum_{k=1}^n \frac{1}{k^2}
= \frac{\pi^2}{6}
\end{displaymath}
```

$$\lim_{n\to\infty} \sum_{k=1}^{n} \frac{1}{k^2} = \frac{\pi^2}{6}$$

數學模式和**文本模式**都一些不同之處。例如，在**數**學模式中：

1. 大多數的空格和斷行沒有任何意義，而且所有的空隙要麼是從相應數學表達式中自然的生成，要麼是用一些專門的命令來指定，如 `\,`， `\quad` 或 `\qquad`。
2. 空白行是不允許的。每個公式只能爲一段。
3. 每一個字母都會被認爲是一個變量名，且會相應被排版爲此種樣式。如果你想要在公式中排版普通的文本（直立字體和普通字距），那麼你必須要把這些文本放在 `\textrm{...}` 命令中（參閱第 45 頁的第 3.7 節）。

```
\begin{equation}
\forall x \in \mathbf{R}:
\qquad x^{2} \geq 0
\end{equation}
```

$$\forall x \in \mathbf{R}: \qquad x^2 \geq 0 \tag{3.2}$$

```
\begin{equation}
x^{2} \geq 0\qquad
\textrm{for all }x\in\mathbf{R}
\end{equation}
```

$$x^2 \geq 0 \qquad \text{for all } x \in \mathbf{R} \tag{3.3}$$

數學家對於符號的使用總是吹毛求疵：這裡習慣上要使用空心粗體 (“blackboard bold”)，要包含此字體，得用到 amsfonts 或是 amssymb 宏包的 \mathbb 命令。上面的例子就變成

```
\begin{displaymath}
x^{2} \geq 0\qquad
\textrm{for all }x\in\mathbb{R}
\end{displaymath}
```

$$x^2 \geq 0 \qquad \text{for all } x \in \mathbb{R}$$

3.2 數學模式的群組

大部分數學模式的命令只對其後的一個字符有效，因此，如果你希望一個命令對多個字符起作用，你必須把它們放在一個群組中，使用花括號：{...}.

```
\begin{equation}
a^x+y \neq a^{x+y}
\end{equation}
```

$$a^x + y \neq a^{x+y} \tag{3.4}$$

3.3 數學公式的基本元素

這一節將介紹數學排版中的最重要的一些命令。詳細的數學排版符號的命令列表，可參閱第 48 頁第 3.10 節。

小寫希臘字母 (Greek letters) 的輸入爲 \alpha、\beta、 \gamma⋯⋯，大寫字母的輸入爲 \Gamma、 \Delta ⋯⋯[2]

```
$\lambda,\xi,\pi,\mu,\Phi,\Omega$
```

$\lambda, \xi, \pi, \mu, \Phi, \Omega$

指數和下標可以能過使用 ^ 和 _ 兩個符號來指定。

```
$a_{1}$ \qquad $x^{2}$ \qquad
$e^{-\alpha t}$ \qquad
$a^{3}_{ij}$\\
$e^{x^2} \neq {e^x}^2$
```

$a_1 \qquad x^2 \qquad e^{-\alpha t} \qquad a^3_{ij}$
$e^{x^2} \neq {e^x}^2$

平方根 (square root) 輸入用 \sqrt；n 次根用 \sqrt[n] 來得到。根號的大小由 LaTeX自動決定。如果僅僅需要根號，可以用 \surd 得到。

```
$\sqrt{x}$ \qquad
$\sqrt{ x^{2}+\sqrt{y} }$
\qquad $\sqrt[3]{2}$\\[3pt]
$\surd[x^2 + y^2]$
```

$\sqrt{x} \qquad \sqrt{x^2+\sqrt{y}} \qquad \sqrt[3]{2}$
$\surd[x^2 + y^2]$

命令 \overline 和 \underline 產生水平線，它們會被放在表達式的正上方或是正下方。

[2] LaTeX 2ε 中沒有定義大寫的 Alpha，因爲它外形與羅馬字母 A 一樣。等到新的數學編碼完成後，情形可能會有所更改。

```
$\overline{m+n}$
```

$\overline{m+n}$

命令 \overbrace 和 \underbrace 可以在一個表達式的上方或下方生成水平括號

```
$\underbrace{a+b+\cdots+z}_{26}$
```

$\underbrace{a+b+\cdots+z}_{26}$

爲了給變量增加數學重音符號，如小箭頭或是~(tilde)，你可以使用第 48 頁表 3.1 所列出的命令。覆蓋多個字符的寬「帽子」和寬~號，可以由 \widehat 和 \widetilde 得到。’ 符號則給出了一個撇號 (prime)。

```
\begin{displaymath}
y=x^{2}\qquad y'=2x\qquad y''=2
\end{displaymath}
```

$$y=x^{2}\qquad y'=2x\qquad y''=2$$

向量可以通過在一個變量上方添加小箭頭 (arrow symbols) 來指定。爲此，使用 \vec 命令即可。\overrightarrow 和 \overleftarrow 這兩個命令可以用來表示一個從 A 到 B 的向量。

```
\begin{displaymath}
\vec a\quad\overrightarrow{AB}
\end{displaymath}
```

$$\vec a\quad\overrightarrow{AB}$$

通常你沒有必要打出一個明顯的點號來表明乘法運算；但是有時候也需要它來幫助讀者分清一個公式。在這些情況下，你應該使用 \cdot 命令。

```
\begin{displaymath}
v = {\sigma}_1 \cdot {\sigma}_2
    {\tau}_1 \cdot {\tau}_2
\end{displaymath}
```

$$v = {\sigma}_1 \cdot {\sigma}_2 {\tau}_1 \cdot {\tau}_2$$

log 等類似的函數名通常是用直立字體，而不是如同變量一樣用斜體，因此 LaTeX 提供了以下的命令來排版這些最重要的函數名：

```
\arccos  \cos   \csc  \exp  \ker    \limsup
\arcsin  \cosh  \deg  \gcd  \lg     \ln
\arctan  \cot   \det  \hom  \lim    \log
\arg     \coth  \dim  \inf  \liminf \max
\sinh    \sup   \tan  \tanh \min    \Pr
\sec     \sin
```

```
\[\lim_{x \rightarrow 0}
\frac{\sin x}{x}=1\]
```

$$\lim_{x \rightarrow 0} \frac{\sin x}{x}=1$$

對於取模函數 (modulo function)，有兩個命令：\bmod 用於二元運算 “$a \bmod b$”，而 \pmod 則用於表達式如 “$x \equiv a \pmod b$”。

```
$a\bmod b$\\
$x\equiv a \pmod{b}$
```

$a \bmod b$
$x \equiv a \pmod{b}$

一個上下的**分式** (fraction) 可用 `\frac{...}{...}` 命令得到。而其傾斜形式如 1/2，有時是更好的選擇，因爲對於簡短的分子分母來說，這看上去更美觀。

```
$1\frac{1}{2}$ hours
\begin{displaymath}
\frac{ x^{2} }{ k+1 }\qquad
x^{ \frac{2}{k+1} }\qquad
x^{ 1/2 }
\end{displaymath}
```

$1\frac{1}{2}$ hours

$$\frac{x^2}{k+1} \qquad x^{\frac{2}{k+1}} \qquad x^{1/2}$$

排版二項式係數或類似的結構，你可以使用 amsmath 宏包中的 `\binom` 命令。

```
\begin{displaymath}
\binom{n}{k}\qquad\mathrm{C}_n^k
\end{displaymath}
```

$$\binom{n}{k} \qquad \mathrm{C}_n^k$$

對於二元關係，有時候你需要到把符號互相堆積起來。 `\stackrel` 命令會把其第一個參數中的符號以上標大小放在第二個上面，而第二個符號則以正常的位置擺放。

```
\begin{displaymath}
\int f_N(x) \stackrel{!}{=} 1
\end{displaymath}
```

$$\int f_N(x) \stackrel{!}{=} 1$$

積分號 (integral operator) 可以用 `\int` 產生，**求和號** (sum operator) 用 `\sum` 命令，而**乘積號** (product operator) 要用 `\prod` 命令。上限和下限用 `^` 和 `_` 來指定，如同上標與下標一樣 [3]。

```
\begin{displaymath}
\sum_{i=1}^{n} \qquad
\int_{0}^{\frac{\pi}{2}} \qquad
\prod_\epsilon
\end{displaymath}
```

$$\sum_{i=1}^{n} \qquad \int_{0}^{\frac{\pi}{2}} \qquad \prod_{\epsilon}$$

爲了更好的控制一個複雜表達式中指標的放置，amsmath 提供了兩個額外的工具： `\substack` 命令和 `subarray` 環境：

```
\begin{displaymath}
\sum_{\substack{0<i<n \\ 1<j<m}}
   P(i,j) =
\sum_{\begin{subarray}{l}
         i\in I\\
         1<j<m
      \end{subarray}}     Q(i,j)
\end{displaymath}
```

$$\sum_{\substack{0<i<n \\ 1<j<m}} P(i,j) = \sum_{\begin{subarray}{l} i\in I \\ 1<j<m \end{subarray}} Q(i,j)$$

[3] $\mathcal{AMS}$-LaTeX 中另有多行的上標/下標。

TEX提供了各種各樣的符號來得到**括號** (braces) 和其他定界符 (delimiters) (如：⌈ ⟨ ‖ ↕)。圓括號和方括號可以由對應的鍵直接輸入而花括號要用 \{，但是所有其它的定界符都要用一定的命令 (如： \updownarrow) 生成。所有可用定界符的列表，請查閱第 50 頁表 3.7。

```
\begin{displaymath}
{a,b,c}\neq\{a,b,c\}
\end{displaymath}
```

$$a, b, c \neq \{a, b, c\}$$

如果你在某個左定界符前放一個 \left 命令或是在某個右定界符前放一個 \right 命令，TEX 將會自動決定這對定界符的大小。請注意，你必須爲每個 \left 命令配對相應的 \right 命令，而且只有在左右定界符被排版在同一行時才會獲得正確的大小尺寸。如果你不想使用任何右定界符，使用看不見的 '\right.' 即可！

```
\begin{displaymath}
1 + \left( \frac{1}{ 1-x^{2} }
    \right) ^3
\end{displaymath}
```

$$1 + \left(\frac{1}{ 1-x^{2} } \right) ^3$$

有些情況下，有必要手工指定一個數學定界符的正確尺寸，這可以使用 \big，\Big，\bigg 和 \Bigg 命令，大多數情況下你只需把它們放在定界符命令的前面[4]。

```
$\Big( (x+1) (x-1) \Big) ^{2}$\\
$\big(\Big(\bigg(\Bigg($\quad
$\big\}\Big\}\bigg\}\Bigg\}$
\quad
$\big\|\Big\|\bigg\|\Bigg\|$
```

$$\Big((x+1) (x-1) \Big) ^{2}$$
$$\big(\Big(\bigg(\Bigg(\quad \big\}\Big\}\bigg\}\Bigg\} \quad \big\|\Big\|\bigg\|\Bigg\|$$

有很多命令可以實現在公式中插入**三點列** (three dots)。\ldots 得到在基線上的點列而 \cdots 是上下居中的點列。另外，還有 \vdots 命令產生豎直的點列，\ddots 產生對角線的點列。 你可以在第 3.5 節找到另外一個例子。

```
\begin{displaymath}
x_{1},\ldots,x_{n} \qquad
x_{1}+\cdots+x_{n}
\end{displaymath}
```

$$x_1, \ldots, x_n \qquad x_1 + \cdots + x_n$$

3.4 數學空格

如果公式內由 TEX 選擇的空格不令人滿意，那麼也可以通過插入一些特殊的空格控制命令來調整。有一些命令可以產生小空格：\, 得到 $\frac{3}{18}$ quad (␣)，\: 得到 $\frac{4}{18}$ quad (␣) 而 \; 會得到 $\frac{5}{18}$ quad (␣)。轉義的空格符 \ 產生一個中等大小的空格，而 \quad (⌴) 和 \qquad (⌴⌴) 產生大的空格。\quad 的大小與當前字體中字母 'M' 的寬度有關。 \! 命令會產生一個 $-\frac{3}{18}$ quad (␣) 的負空格。

[4]如果使用了某個改變字體大小的命令，或是指定了 11pt 或 12pt 參數的話，這些命令會達不到預期效果。使用 exscale 或 amsmath 宏包可以糾正它。

```
\newcommand{\ud}{\mathrm{d}}
\begin{displaymath}
\int\!\!\!\int_{D} g(x,y)
  \, \ud x\, \ud y
\end{displaymath}
instead of
\begin{displaymath}
\int\int_{D} g(x,y)\ud x \ud y
\end{displaymath}
```

$$\iint_D g(x,y)\,\mathrm{d}x\,\mathrm{d}y$$

instead of

$$\int\int_D g(x,y)\mathrm{d}x\mathrm{d}y$$

請注意這裡微分中的 'd' 按慣例要設定成羅馬字體。

$\mathcal{AMS}$-LaTeX 爲多重積分號之間空格的微調提供了另一種方法，即使用 `\iint`, `\iiint`, `\iiiint`, 和 `\idotsint` 命令。

加入 amsmath 宏包後，上面的例子可以寫成這樣：

```
\newcommand{\ud}{\mathrm{d}}
\begin{displaymath}
\iint_{D} \, \ud x \, \ud y
\end{displaymath}
```

$$\iint_D \mathrm{d}x\,\mathrm{d}y$$

更多詳情請參見電子文檔 testmath.tex（與 $\mathcal{AMS}$-LaTeX一起發行）或 *The LaTeX Companion* [3] 的第八章。

3.5 垂直取齊

要排版數組，使用 `array` 環境。它的使用與 `tabular` 環境有些類似。 `\\` 命令可用來斷行。

```
\begin{displaymath}
\mathbf{X} =
\left( \begin{array}{ccc}
x_{11} & x_{12} & \ldots \\
x_{21} & x_{22} & \ldots \\
\vdots & \vdots & \ddots
\end{array} \right)
\end{displaymath}
```

$$\mathbf{X} = \left(\begin{array}{ccc} x_{11} & x_{12} & \ldots \\ x_{21} & x_{22} & \ldots \\ \vdots & \vdots & \ddots \end{array} \right)$$

`array` 環境也可以用來排版這樣的表達式，表達式中使用一個 "." 作爲其隱藏的 `\right` 定界符。

```
\begin{displaymath}
y = \left\{ \begin{array}{ll}
 a & \textrm{if $d>c$}\\
 b+x & \textrm{in the morning}\\
 l & \textrm{all day long}
  \end{array} \right.
\end{displaymath}
```

$$y = \left\{ \begin{array}{ll} a & \textrm{if } d > c \\ b + x & \textrm{in the morning} \\ l & \textrm{all day long} \end{array} \right.$$

就像在 `tabular` 環境中一樣，你也可以在 `array` 環境中畫線，如分隔矩陣中元素：

```
\begin{displaymath}
\left(\begin{array}{c|c}
 1 & 2 \\
\hline
3 & 4
\end{array}\right)
\end{displaymath}
```

$$\left(\begin{array}{c|c} 1 & 2 \\ \hline 3 & 4 \end{array}\right)$$

對於跨行的長公式或是方程組 (equation system)，你可以使用 `eqnarray` 和 `eqnarray*` 環境來替代 `equation` 環境。在 `eqnarray` 環境中每一行都有一個等式編號。`eqnarray*` 則不添加編號。

`eqnarray` 和 `eqnarray*` 環境的用法與一個 `{rcl}` 形式的 3 列表格相類似，這裡中間一列可以用來放等號，不等號，或者是其他你選擇的符號。 `\\` 命令可以斷行。

```
\begin{eqnarray}
f(x) & = & \cos x     \\
f'(x) & = & -\sin x   \\
\int_{0}^{x} f(y)dy &
 = & \sin x
\end{eqnarray}
```

$$\begin{aligned} f(x) &= \cos x && (3.5)\\ f'(x) &= -\sin x && (3.6)\\ \int_0^x f(y)dy &= \sin x && (3.7) \end{aligned}$$

注意，這裡等號兩邊空白都有些大。`\setlength\arraycolsep{2pt}` 可以調小它，比如在下一個例子裡。

長等式不能被分成合適的小段。作者必須指定在哪裡斷且如何縮進。以下兩種方法是最常用的。

```
{\setlength\arraycolsep{2pt}
\begin{eqnarray}
\sin x & = & x -\frac{x^{3}}{3!}
     +\frac{x^{5}}{5!}-{}
                     \nonumber\\
&& {}-\frac{x^{7}}{7!}+{}\cdots
\end{eqnarray}}
```

$$\begin{aligned} \sin x &= x - \frac{x^3}{3!} + \frac{x^5}{5!} - \\ &\quad - \frac{x^7}{7!} + \cdots && (3.8) \end{aligned}$$

```
\begin{eqnarray}
\lefteqn{ \cos x = 1
     -\frac{x^{2}}{2!} +{} }
                     \nonumber\\
 & & {}+\frac{x^{4}}{4!}
     -\frac{x^{6}}{6!}+{}\cdots
\end{eqnarray}
```

$$\begin{aligned} \cos x &= 1 - \frac{x^2}{2!} + \\ &\quad + \frac{x^4}{4!} - \frac{x^6}{6!} + \cdots && (3.9) \end{aligned}$$

`\nonumber` 命令告訴 LaTeX 不要給這個等式編號。

用這種方法很難讓等式正確的垂直對齊；amsmath 宏包提供了一系列強有力的替代選擇（參見 `align`, `flalign`, `gather`, `multline` 和 `split` 環境）。

3.6 虛位

我們看不見虛位（phantom，也有幻影的意思），但是在許多人的頭腦中它們依然佔有一定的位置。LaTeX 中也一樣。我們可以使用它來實現一些有趣的小技巧。

當使用 ^ 和 _ 時，LaTeX 對文本的垂直對齊有時顯得太過於自作多情。使用 \phantom 命令你可以給不在最終輸出中顯示的字符保留位置。理解此意的最好方法是看下面的例子。

```
\begin{displaymath}
{}^{12}_{\phantom{1}6}\textrm{C}
\qquad \textrm{versus} \qquad
{}^{12}_{6}\textrm{C}
\end{displaymath}
```

$$ {}^{12}_{6}\mathrm{C} \qquad \text{versus} \qquad {}^{12}_{6}\mathrm{C} $$

```
\begin{displaymath}
\Gamma_{ij}^{\phantom{ij}k}
\qquad \textrm{versus} \qquad
\Gamma_{ij}^{k}
\end{displaymath}
```

$$ \Gamma_{ij}^{k} \qquad \text{versus} \qquad \Gamma_{ij}^{k} $$

3.7 數學字體尺寸

在數學模式中，TeX 根據上下文選擇字體大小。例如，上標會排版成較小的字體。如果你想要把等式的一部分排版成羅馬字體，不要用 \textrm 命令，只因 \textrm 會暫時切換到文本模式，而此時字體大小切換機制將不起作用。使用 \mathrm 來保持字體大小切換機制的正常。但是要小心，\mathrm 只對較短的項有效。空格依然無效而且重音符號也不起作用[5]。

```
\begin{equation}
2^{\textrm{nd}} \quad
2^{\mathrm{nd}}
\end{equation}
```

$$ 2^{\textrm{nd}} \quad 2^{\mathrm{nd}} \tag{3.10} $$

有時你仍需告訴 LaTeX 正確的字體大小。在數學模式中，可用以下四個命令來設定：

\displaystyle (123), \textstyle (123), \scriptstyle (123) and \scriptscriptstyle (123).

改變樣式也會影響到上下限的顯示方式。

```
\begin{displaymath}
 \frac{\displaystyle
   \sum_{i=1}^n(x_i-\overline x)
   (y_i-\overline y)}
  {\displaystyle\biggl[
 \sum_{i=1}^n(x_i-\overline x)^2
\sum_{i=1}^n(y_i-\overline y)^2
\biggr]^{1/2}}
\end{displaymath}
```

$$ \frac{\displaystyle \sum_{i=1}^{n}(x_i-\overline{x})(y_i-\overline{y})}{\displaystyle \left[\sum_{i=1}^{n}(x_i-\overline{x})^2\sum_{i=1}^{n}(y_i-\overline{y})^2\right]^{1/2}} $$

[5] $\mathcal{AMS}$-LaTeX(amsmath) 宏包可以讓 \textrm 命令與字體大小切換機制和諧共存。

這個例子中的括號要比 \left[\right] 提供的括號更大些。\biggl 和 \biggr 命令分別對應於左和右括號。

3.8 定理、定律……

當寫數學文檔時，你可能需要一種方法來排版「引理」、「定義」、「公理」及其他類似的結構。

\newtheorem{*name*}[*counter*]{*text*}[*section*]

參量 *name* 是用來標識「定理」的短關鍵字。而參數 *text* 才是眞正的「定理」名，它會在最終的文檔中被列印出來。

方括號中是可選參量。兩者都均用來指定「定理」的編號問題。使用 *counter* 參數來指定先前聲明的「定理」的 *name*。則此新的「定理」將與先前定理統一編號。*section* 參數讓你來指定章節單元，而「定理」會按相應的章節層次來編號。

在你的文檔的導言區執行 \newtheorem 命令後，你就可以在文檔中使用以下命令了。

\begin{*name*}[*text*]
This is my interesting theorem
\end{*name*}

amsthm 宏包提供了 \newtheoremstyle{*style*} 命令，通過從三個預定義樣式中選擇其一來定義定理的外觀，三個樣式分別爲：definition （標題粗體，內容羅馬體），plain （標題粗體，內容斜體）和 remark （標題斜體，內容羅馬體）。

理論上已經說夠多了，下面我們聯繫一下實踐，這個例子希望能夠帶走你的疑問並讓你知道 \newtheorem 環境其實比較複雜且不易理解。

首先定義定理環境

```
\theoremstyle{definition} \newtheorem{law}{Law}
\theoremstyle{plain}      \newtheorem{jury}[law]{Jury}
\theoremstyle{remark}     \newtheorem*{marg}{Margaret}
```

```
\begin{law} \label{law:box}
Don't hide in the witness box
\end{law}
\begin{jury}[The Twelve]
It could be you! So beware and
see law \ref{law:box}\end{jury}
\begin{marg}No, No, No\end{marg}
```

Law 1. Don't hide in the witness box

Jury 2 (The Twelve)**.** *It could be you! So beware and see law 1*

Margaret. No, No, No

「Jury」定理與「Law」定理共用了同一個計數器，因此它的編號與其他「Law」定理的編號是順序下來的。方括號中的參量用來指定定理的一個標題或是其他類似的內容。

```
\flushleft
\newtheorem{mur}{Murphy}[section]
\begin{mur}
If there are two or more
ways to do something, and
one of those ways can result
in a catastrophe, then
someone will do it.\end{mur}
```

Murphy 3.8.1. If there are two or more ways to do something, and one of those ways can result in a catastrophe, then someone will do it.

「 Murphy 」定理有一個與當前章節相聯繫的編號。你也可以使用其他的單元，如章 (chapter) 或小節 (subsection)。

amsthm 還提供了一個 `proof` 環境。

```
\begin{proof}
 Trivial, use
\[E=mc^2\]
\end{proof}
```

Proof. Trivial, use

$$E = mc^2$$

□

使用 `\qedhere` 命令你可以移動「 證畢 」符。「 證畢 」符默認是在證明結束時單獨放於一行。

```
\begin{proof}
 Trivial, use
\[E=mc^2 \qedhere\]
\end{proof}
```

Proof. Trivial, use

$$E = mc^2 \qquad \square$$

3.9 粗體符號

在 LaTeX 中要得到粗體符號相當的不容易；這也許是故意設置的，以防業餘水平的排版者過度的使用它們。字體變換命令 `\mathbf` 可得到粗體字母，但是得到的是羅馬體 (直立的) 而數學符號通常要求是斜體。還有一個 `\boldmath` 命令，但是它只能用在數學模式之外。它不僅作用於字母也作用於符號。

```
\begin{displaymath}
\mu, M \qquad \mathbf{M} \qquad
\mbox{\boldmath $\mu, M$}
\end{displaymath}
```

$$\mu, M \qquad \mathbf{M} \qquad \boldsymbol{\mu, M}$$

請注意，逗號也成粗體了，這也許不是所需的。

使用 amsbsy 宏包 (包含在 amsmath 中) 或 tool 宏包集中的 bm 將會便利許多，因爲它們包含一個叫 `\boldsymbol` 的命令。

```
\begin{displaymath}
\mu, M \qquad
\boldsymbol{\mu}, \boldsymbol{M}
\end{displaymath}
```

$$\mu, M \qquad \boldsymbol{\mu, M}$$

3.10 數學符號表

以下表格列出了**數學模式**中的所有常用符號。

要使用表 3.11–3.15[6]。必須在導言區先載入 amssymb 宏包而且系統中安裝了 AMS 數學字體。如果係統中沒有安裝 AMS 宏包和字體，請查閱 `macros/latex/required/amslatex`。更全面的列表可於 `info/symbols/comprehensive` 處找到。

表 3.1 – 數學模式重音符號。

$\hat{a}$	`\hat{a}`	$\check{a}$	`\check{a}`	$\tilde{a}$	`\tilde{a}`
$\grave{a}$	`\grave{a}`	$\dot{a}$	`\dot{a}`	$\ddot{a}$	`\ddot{a}`
$\bar{a}$	`\bar{a}`	$\vec{a}$	`\vec{a}`	$\widehat{A}$	`\widehat{A}`
$\acute{a}$	`\acute{a}`	$\breve{a}$	`\breve{a}`	$\widetilde{A}$	`\widetilde{A}`

表 3.2 – 希臘字母。

α	`\alpha`	θ	`\theta`	o	`o`	υ	`\upsilon`
β	`\beta`	ϑ	`\vartheta`	π	`\pi`	ϕ	`\phi`
γ	`\gamma`	ι	`\iota`	ϖ	`\varpi`	φ	`\varphi`
δ	`\delta`	κ	`\kappa`	ρ	`\rho`	χ	`\chi`
ϵ	`\epsilon`	λ	`\lambda`	ϱ	`\varrho`	ψ	`\psi`
ε	`\varepsilon`	μ	`\mu`	σ	`\sigma`	ω	`\omega`
ζ	`\zeta`	ν	`\nu`	ς	`\varsigma`		
η	`\eta`	ξ	`\xi`	τ	`\tau`		
Γ	`\Gamma`	Λ	`\Lambda`	Σ	`\Sigma`	Ψ	`\Psi`
Δ	`\Delta`	Ξ	`\Xi`	Υ	`\Upsilon`	Ω	`\Omega`
Θ	`\Theta`	Π	`\Pi`	Φ	`\Phi`		

[6]這些表格源自於 David Carlisle 的 `symbols.tex`, 而後在 Josef Tkadlec 的建議下作了較大的改動。

表 3.3 – 二元關係。

你可以在下列符號的相應命令前加上 \not 命令，而得到其否定形式。

$<$	<	$>$	>	$=$	=
$\leq$	\leq or \le	$\geq$	\geq or \ge	$\equiv$	\equiv
$\ll$	\ll	$\gg$	\gg	$\doteq$	\doteq
$\prec$	\prec	$\succ$	\succ	$\sim$	\sim
$\preceq$	\preceq	$\succeq$	\succeq	$\simeq$	\simeq
$\subset$	\subset	$\supset$	\supset	$\approx$	\approx
$\subseteq$	\subseteq	$\supseteq$	\supseteq	$\cong$	\cong
$\sqsubset$ [a]	\sqsubset [a]	$\sqsupset$ [a]	\sqsupset [a]	$\bowtie$ [a]	\Join [a]
$\sqsubseteq$	\sqsubseteq	$\sqsupseteq$	\sqsupseteq	$\bowtie$	\bowtie
$\in$	\in	$\ni$	\ni , \owns	$\propto$	\propto
$\vdash$	\vdash	$\dashv$	\dashv	$\models$	\models
$\mid$	\mid	$\parallel$	\parallel	$\perp$	\perp
$\smile$	\smile	$\frown$	\frown	$\asymp$	\asymp
$:$	:	$\notin$	\notin	$\neq$	\neq or \ne

[a] 使用 latexsym 宏包才能得到這個符號

表 3.4 – 二元運算符。

$+$	+	$-$	-		
$\pm$	\pm	$\mp$	\mp	$\triangleleft$	\triangleleft
$\cdot$	\cdot	$\div$	\div	$\triangleright$	\triangleright
$\times$	\times	$\setminus$	\setminus	$\star$	\star
$\cup$	\cup	$\cap$	\cap	$*$	\ast
$\sqcup$	\sqcup	$\sqcap$	\sqcap	$\circ$	\circ
$\vee$	\vee , \lor	$\wedge$	\wedge , \land	$\bullet$	\bullet
$\oplus$	\oplus	$\ominus$	\ominus	$\diamond$	\diamond
$\odot$	\odot	$\oslash$	\oslash	$\uplus$	\uplus
$\otimes$	\otimes	$\bigcirc$	\bigcirc	$\amalg$	\amalg
$\bigtriangleup$	\bigtriangleup	$\bigtriangledown$	\bigtriangledown	$\dagger$	\dagger
$\lhd$ [a]	\lhd [a]	$\rhd$ [a]	\rhd [a]	$\ddagger$	\ddagger
$\unlhd$ [a]	\unlhd [a]	$\unrhd$ [a]	\unrhd [a]	$\wr$	\wr

表 3.5 –「大」運算符。

$\sum$	\sum	$\bigcup$	\bigcup	$\bigvee$	\bigvee
$\prod$	\prod	$\bigcap$	\bigcap	$\bigwedge$	\bigwedge
$\coprod$	\coprod	$\bigsqcup$	\bigsqcup	$\biguplus$	\biguplus
$\int$	\int	$\oint$	\oint	$\bigodot$	\bigodot
$\bigoplus$	\bigoplus	$\bigotimes$	\bigotimes		

表 3.6 – 箭頭。

$\leftarrow$	\leftarrow or \gets	$\longleftarrow$	\longleftarrow
$\rightarrow$	\rightarrow or \to	$\longrightarrow$	\longrightarrow
$\leftrightarrow$	\leftrightarrow	$\longleftrightarrow$	\longleftrightarrow
$\Leftarrow$	\Leftarrow	$\Longleftarrow$	\Longleftarrow
$\Rightarrow$	\Rightarrow	$\Longrightarrow$	\Longrightarrow
$\Leftrightarrow$	\Leftrightarrow	$\Longleftrightarrow$	\Longleftrightarrow
$\mapsto$	\mapsto	$\longmapsto$	\longmapsto
$\hookleftarrow$	\hookleftarrow	$\hookrightarrow$	\hookrightarrow
$\leftharpoonup$	\leftharpoonup	$\rightharpoonup$	\rightharpoonup
$\leftharpoondown$	\leftharpoondown	$\rightharpoondown$	\rightharpoondown
$\rightleftharpoons$	\rightleftharpoons	$\iff$	\iff (bigger spaces)
$\uparrow$	\uparrow	$\downarrow$	\downarrow
$\updownarrow$	\updownarrow	$\Uparrow$	\Uparrow
$\Downarrow$	\Downarrow	$\Updownarrow$	\Updownarrow
$\nearrow$	\nearrow	$\searrow$	\searrow
$\swarrow$	\swarrow	$\nwarrow$	\nwarrow
$\leadsto$	\leadsto [a]		

[a] 使用 latexsym 宏包才能得到這個符號

表 3.7 – 定界符。

(	(	)	)	$\uparrow$	\uparrow
[	[or \lbrack	]	] or \rbrack	$\downarrow$	\downarrow
{	\{ or \lbrace	}	\} or \rbrace	$\updownarrow$	\updownarrow
$\langle$	\langle	$\rangle$	\rangle	$\vert$	| or \vert
$\lfloor$	\lfloor	$\rfloor$	\rfloor	$\lceil$	\lceil
/	/	\	\backslash	$\Updownarrow$	\Updownarrow
$\Uparrow$	\Uparrow	$\Downarrow$	\Downarrow	$\Vert$	\\| or \Vert
$\rceil$	\rceil				

表 3.8 – 大定界符。

$\lgroup$	\lgroup	$\rgroup$	\rgroup	$\lmoustache$	\lmoustache
$\arrowvert$	\arrowvert	$\Arrowvert$	\Arrowvert	$\bracevert$	\bracevert
$\rmoustache$	\rmoustache				

表 3.9 – 其他符號。

$\dots$	\dots	$\cdots$	\cdots	$\vdots$	\vdots	$\ddots$	\ddots
$\hbar$	\hbar	$\imath$	\imath	$\jmath$	\jmath	ℓ	\ell
$\Re$	\Re	$\Im$	\Im	$\aleph$	\aleph	$\wp$	\wp
$\forall$	\forall	$\exists$	\exists	$\mho$ [a]	\mho [a]	∂	\partial
$'$	'	$\prime$	\prime	$\emptyset$	\emptyset	∞	\infty
∇	\nabla	$\triangle$	\triangle	$\Box$ [a]	\Box [a]	$\Diamond$ [a]	\Diamond [a]
$\bot$	\bot	$\top$	\top	$\angle$	\angle	$\surd$	\surd
$\diamondsuit$	\diamondsuit	$\heartsuit$	\heartsuit	$\clubsuit$	\clubsuit	$\spadesuit$	\spadesuit
$\neg$	\neg or \lnot	$\flat$	\flat	$\natural$	\natural	$\sharp$	\sharp

[a]使用 latexsym 宏包才能得到這個符號

表 3.10 – 非數學符號。

也可以在文本模式中使用這些符號。

†	\dag	§	\S	©	\copyright	®	\textregistered
‡	\ddag	¶	\P	£	\pounds	%	\%

表 3.11 – AMS 定界符。

⌜	\ulcorner	⌝	\urcorner	⌞	\llcorner	⌟	\lrcorner
$\lvert$	\lvert	$\rvert$	\rvert	$\lVert$	\lVert	$\rVert$	\rVert

表 3.12 – AMS 希臘和希伯來字母。

$\digamma$	\digamma	$\varkappa$	\varkappa	ℶ	\beth	ℷ	\gimel	ℸ	\daleth

表 3.13 – AMS 二元關係。

$\leqslant$	\lessdot	$\geqslant$	\gtrdot	$\doteqdot$	\doteqdot
$\leqslant$	\leqslant	$\geqslant$	\geqslant	$\risingdotseq$	\risingdotseq
$\leqslant$	\eqslantless	$\geqslant$	\eqslantgtr	$\fallingdotseq$	\fallingdotseq
$\leqq$	\leqq	$\geqq$	\geqq	$\eqcirc$	\eqcirc
$\lll$	\lll or \llless	$\ggg$	\ggg	$\circeq$	\circeq
$\lesssim$	\lesssim	$\gtrsim$	\gtrsim	$\triangleq$	\triangleq
$\lessapprox$	\lessapprox	$\gtrapprox$	\gtrapprox	$\bumpeq$	\bumpeq
$\lessgtr$	\lessgtr	$\gtrless$	\gtrless	$\Bumpeq$	\Bumpeq
$\lesseqgtr$	\lesseqgtr	$\gtreqless$	\gtreqless	$\thicksim$	\thicksim
$\lesseqqgtr$	\lesseqqgtr	$\gtreqqless$	\gtreqqless	$\thickapprox$	\thickapprox
$\preccurlyeq$	\preccurlyeq	$\succcurlyeq$	\succcurlyeq	$\approxeq$	\approxeq
$\curlyeqprec$	\curlyeqprec	$\curlyeqsucc$	\curlyeqsucc	$\backsim$	\backsim
$\precsim$	\precsim	$\succsim$	\succsim	$\backsimeq$	\backsimeq
$\precapprox$	\precapprox	$\succapprox$	\succapprox	$\vDash$	\vDash
$\subseteqq$	\subseteqq	$\supseteqq$	\supseteqq	$\Vdash$	\Vdash
$\shortparallel$	\shortparallel	$\Supset$	\Supset	$\Vvdash$	\Vvdash
$\blacktriangleleft$	\blacktriangleleft	$\sqsupset$	\sqsupset	$\backepsilon$	\backepsilon
$\vartriangleright$	\vartriangleright	$\because$	\because	$\varpropto$	\varpropto
$\blacktriangleright$	\blacktriangleright	$\Subset$	\Subset	$\between$	\between
$\trianglerighteq$	\trianglerighteq	$\smallfrown$	\smallfrown	$\pitchfork$	\pitchfork
$\vartriangleleft$	\vartriangleleft	$\shortmid$	\shortmid	$\smallsmile$	\smallsmile
$\trianglelefteq$	\trianglelefteq	$\therefore$	\therefore	$\sqsubset$	\sqsubset

表 3.14 – AMS 箭頭。

$\dashleftarrow$	\dashleftarrow	$\dashrightarrow$	\dashrightarrow
$\leftleftarrows$	\leftleftarrows	$\rightrightarrows$	\rightrightarrows
$\leftrightarrows$	\leftrightarrows	$\rightleftarrows$	\rightleftarrows
$\Lleftarrow$	\Lleftarrow	$\Rrightarrow$	\Rrightarrow
$\twoheadleftarrow$	\twoheadleftarrow	$\twoheadrightarrow$	\twoheadrightarrow
$\leftarrowtail$	\leftarrowtail	$\rightarrowtail$	\rightarrowtail
$\leftrightharpoons$	\leftrightharpoons	$\rightleftharpoons$	\rightleftharpoons
$\Lsh$	\Lsh	$\Rsh$	\Rsh
$\looparrowleft$	\looparrowleft	$\looparrowright$	\looparrowright
$\curvearrowleft$	\curvearrowleft	$\curvearrowright$	\curvearrowright
$\circlearrowleft$	\circlearrowleft	$\circlearrowright$	\circlearrowright
$\multimap$	\multimap	$\upuparrows$	\upuparrows
$\downdownarrows$	\downdownarrows	$\upharpoonleft$	\upharpoonleft
$\upharpoonright$	\upharpoonright	$\downharpoonright$	\downharpoonright
$\rightsquigarrow$	\rightsquigarrow	$\leftrightsquigarrow$	\leftrightsquigarrow

表 3.15 – AMS 二元否定關係符和箭頭。

$\nless$	\nless	$\ngtr$	\ngtr	$\varsubsetneqq$	\varsubsetneqq
$\lneq$	\lneq	$\gneq$	\gneq	$\varsupsetneqq$	\varsupsetneqq
$\nleq$	\nleq	$\ngeq$	\ngeq	$\nsubseteqq$	\nsubseteqq
$\nleqslant$	\nleqslant	$\ngeqslant$	\ngeqslant	$\nsupseteqq$	\nsupseteqq
$\lneqq$	\lneqq	$\gneqq$	\gneqq	$\nmid$	\nmid
$\lvertneqq$	\lvertneqq	$\gvertneqq$	\gvertneqq	$\nparallel$	\nparallel
$\nleqq$	\nleqq	$\ngeqq$	\ngeqq	$\nshortmid$	\nshortmid
$\lnsim$	\lnsim	$\gnsim$	\gnsim	$\nshortparallel$	\nshortparallel
$\lnapprox$	\lnapprox	$\gnapprox$	\gnapprox	$\nsim$	\nsim
$\nprec$	\nprec	$\nsucc$	\nsucc	$\ncong$	\ncong
$\npreceq$	\npreceq	$\nsucceq$	\nsucceq	$\nvdash$	\nvdash
$\precneqq$	\precneqq	$\succneqq$	\succneqq	$\nvDash$	\nvDash
$\precnsim$	\precnsim	$\succnsim$	\succnsim	$\nVdash$	\nVdash
$\precnapprox$	\precnapprox	$\succnapprox$	\succnapprox	$\nVDash$	\nVDash
$\subsetneq$	\subsetneq	$\supsetneq$	\supsetneq	$\ntriangleleft$	\ntriangleleft
$\varsubsetneq$	\varsubsetneq	$\varsupsetneq$	\varsupsetneq	$\ntriangleright$	\ntriangleright
$\nsubseteq$	\nsubseteq	$\nsupseteq$	\nsupseteq	$\ntrianglelefteq$	\ntrianglelefteq
$\subsetneqq$	\subsetneqq	$\supsetneqq$	\supsetneqq	$\ntrianglerighteq$	\ntrianglerighteq
$\nleftarrow$	\nleftarrow	$\nrightarrow$	\nrightarrow	$\nleftrightarrow$	\nleftrightarrow
$\nLeftarrow$	\nLeftarrow	$\nRightarrow$	\nRightarrow	$\nLeftrightarrow$	\nLeftrightarrow

表 3.16 – AMS 二元運算符。

$\dotplus$	\dotplus	$\centerdot$	\centerdot		
$\ltimes$	\ltimes	$\rtimes$	\rtimes	$\divideontimes$	\divideontimes
$\Cup$	\doublecup	$\Cap$	\doublecap	$\smallsetminus$	\smallsetminus
$\veebar$	\veebar	$\barwedge$	\barwedge	$\doublebarwedge$	\doublebarwedge
$\boxplus$	\boxplus	$\boxminus$	\boxminus	$\circleddash$	\circleddash
$\boxtimes$	\boxtimes	$\boxdot$	\boxdot	$\circledcirc$	\circledcirc
$\intercal$	\intercal	$\circledast$	\circledast	$\rightthreetimes$	\rightthreetimes
$\curlyvee$	\curlyvee	$\curlywedge$	\curlywedge	$\leftthreetimes$	\leftthreetimes

表 3.17 – AMS 其他符號。

$\hbar$	\hbar	$\hslash$	\hslash	$\Bbbk$	\Bbbk
$\square$	\square	$\blacksquare$	\blacksquare	Ⓢ	\circledS
$\vartriangle$	\vartriangle	$\blacktriangle$	\blacktriangle	$\complement$	\complement
$\triangledown$	\triangledown	$\blacktriangledown$	\blacktriangledown	$\Game$	\Game
$\lozenge$	\lozenge	$\blacklozenge$	\blacklozenge	$\bigstar$	\bigstar
$\angle$	\angle	$\measuredangle$	\measuredangle		
$\diagup$	\diagup	$\diagdown$	\diagdown	$\backprime$	\backprime
$\nexists$	\nexists	$\Finv$	\Finv	$\varnothing$	\varnothing
$\eth$	\eth	$\sphericalangle$	\sphericalangle	$\mho$	\mho

表 3.18 – 數學字母。

實例	命令	所需宏包
ABCDEabcde1234	\mathrm{ABCDE abcde 1234}	
$ABCDEabcde1234$	\mathit{ABCDE abcde 1234}	
$ABCDEabcde1234$	\mathnormal{ABCDE abcde 1234}	
$\mathcal{ABCDE}$	\mathcal{ABCDE abcde 1234}	
$\mathscr{ABCDE}$	\mathscr{ABCDE abcde 1234}	mathrsfs
$\mathfrak{ABCDEabcde1234}$	\mathfrak{ABCDE abcde 1234}	amsfonts or amssymb
$\mathbb{ABCDE}\daleth\beth\gimel$	\mathbb{ABCDE abcde 1234}	amsfonts or amssymb

Chapter 4

專業功能

當你整理一個大型文檔時，LaTeX 的一些專門功能，例如自動生成索引、管理參考文獻等等，會給你以很大的幫助。詳細的關於 LaTeX 專業功能以及增強功能的描述可以在 *LaTeX Manual* [1] 和 *The LaTeX Companion* [3] 找到。

4.1 插入 EPS 圖形

LaTeX 通過 `figure` 和 `table` 環境提供了處理圖像圖形等浮動體的基本工具。

有幾種辦法可以通過使用基本 LaTeX 命令或者 LaTeX 擴展宏包來產生實際的圖形 (graphics)，第 5 章中將會介紹其中的幾種方法。如果需要這方面的詳細信息，請參閱 *The LaTeX Companion* [3] 和 *LaTeX Manual* [1]。

在文檔中使用圖形，一個相對容易的辦法就是使用專門的軟件包[1]生成圖形文件，然後將最終的圖形文件插入到文檔中。LaTeX 的宏包提供了許多方法來完成這個工作。在這個手册裡，我們只討論 Encapsulated PostScript (EPS) 圖形文件的使用，因爲它比較簡單而且被廣泛地使用。爲了使用 EPS 格式的圖片，你必須有一個 PostScript列印機 [2]來輸出結果。

由 D. P. Carlisle 製作的 graphicx 宏包提供了一套很好的插圖命令。它是一個叫作 “graphics” 的宏包集中的一部分 [3]。

假設你使用的系統安裝了 PostScript 列印機和 graphicx 宏包，那麼你就可以通過下面的步驟把一幅圖片加入你的文檔中：

1. 用你的圖形軟件輸出 EPS 格式的文件[4]。
2. 在源文件的導言中加上下面的命令來載入 graphicx 宏包。

   `\usepackage[`*driver*`]{graphicx}`

   這裡 *driver* 是你使用的「dvi 到 postscript」的轉換程序。最常用的是 `dvips`。因爲 TeX 中沒有規定插入圖形的標準，所以 *driver* 的名字是

[1] 例如 XFig、 CorelDraw!、 Freehand、 Gnuplot……

[2] 另外一個可以用來輸出 PostScript 的工具是 GhostScript 程序，它可以從 `support/ghostscript` 得到。Windows 和 OS/2 用戶可能更喜歡用 GSview。

[3] `macros/latex/required/graphics`

[4] 如果你的軟件不能輸出 EPS 格式的文件，你可以嘗試安裝一個 PostScript 列印機驅動程序（例如 Apple LaserWriter），然後將你的圖形通過這個驅動程序列印到文件。運氣好的話，這個文件會是 EPS 格式的。注意一個 EPS 圖片不能包含超過一頁的內容。一些列印機驅動程序可以明確地指定輸出 EPS 格式。

必需的。知道了 *driver* 的名字，graphicx 宏包就可以選擇合適的方法在 `.dvi` 文件中插入關於圖形的信息。然後列印機理解這些信息並能正確的插入 `.eps` 文件。

3. 使用命令

   `\includegraphics[`*key*=*value*, ...`]{`*file*`}`

   來把 *file* 加入你的文檔。可選的參數是一系列由逗號隔開的 *keys* 和相應的 (*values*)。*keys* 可以用來改變插圖的寬度、高度以及旋轉。表 4.1 列出了最重要的關鍵詞。

表 4.1 – graphicx 宏包使用的關鍵詞。

`width`	把圖形調整到指定的寬度
`height`	把圖形調整到指定的高度
`angle`	逆時針旋轉圖形
`scale`	縮放圖形

下面的示例代碼可以幫助我們理解整個過程：

```
\begin{figure}
\centering
\includegraphics[angle=90,
                 width=0.5\textwidth]{test}
\caption{This is a test.}
\end{figure}
```

這段代碼把文件 `test.eps` 中的圖形插入到文檔裡。首先圖形被旋轉 90 度，然後進行縮放使得圖形的寬度等於標準段落寬度的 0.5 倍。因爲沒有指定圖形的高度，圖形的高寬變化的比例是 1.0，也就是保持原來的高寬比。高度和寬度參數也可以指定爲絕對長度單位。詳細的信息可以在第 92 頁的表 6.5 中找到。如果你想知道更多這方面的知識，請閱讀文獻 [9] 和 [13]。

4.2 參考文獻

你可以通過 `thebibliography` 環境來產生一個參考文獻 (bibliography)。每個參考文獻的條目以如下的命令開頭

`\bibitem[`*label*`]{`*marker*`}`

然後使用 *marker* 在正文中引用這本書、這篇文章或者論文。

`\cite{`*marker*`}`

如果你不使用參數 *label*，參考文獻條目的編號是自動生成的。`\begin{thebibliography}` 命令後的參數設定了最大的編號寬度。在下面的例子中，`{99}` 告訴 LaTeX 參考文獻條目的編號不會比數字 99 更寬。

```
Partl \cite{pa} has
proposed that \ldots
\begin{thebibliography}{99}
\bibitem{pa} H. Partl:
\emph{German \TeX},
TUGboat Volume 9, Issue 1
 (1988)
\end{thebibliography}
```

Partl [1] has proposed that . . .

Bibliography

[1] H. Partl: *German TEX*, TUGboat Volume 9, Issue 1 (1988)

對於大型項目，你也許需要使用 BibTEX 程序。 BibTEX 包含在大多數的 TEX 發行版本中。它能夠讓你維護一個參考文獻資料庫，並從中生成你的論文引用到的文獻條目。 BibTEX 產生的參考文獻的外觀是基於樣式表，它可以讓你按照大量預先設計好的格式來創建你的參考文獻。

4.3 索引

許多書籍的一個非常有用的部分就是它們的索引 (index) 了。使用 LATEX 和輔助工具 `makeindex`[5]，我們能夠很容易的生成索引。在這個手册裡，只介紹了最基本的索引生成命令。更進一步的瞭解請參考 *The LATEX Companion* [3]。

爲了使用 LATEX 的索引功能，宏包 makeidx 必須在導言部分被載入：

```
\usepackage{makeidx}
```

然後在導言中使用

```
\makeindex
```

啓動索引命令。

索引的內容通過命令

```
\index{key}
```

指定，這裡 *key* 是索引項的關鍵詞。你可以在需要被索引的地方加入這條命令。表 4.2 舉例解釋了參量 *key* 語法。

當 LATEX 處理源文檔時，每個 `\index` 命令都會將適當的索引項和當前頁碼寫入一個專門的文件中。這個文件的名字和 LATEX 源文檔相同，但具有不同

[5]在文件名不能超過 8 個字符的操作系統上，這個程序被命名爲 `makeidx`。

表 4.2 – 索引關鍵詞語法示例。

示例	索引項	註釋
`\index{hello}`	hello, 1	普通格式的索引項
`\index{hello!Peter}`	Peter, 3	'hello' 下的子項
`\index{Sam@\textsl{Sam}}`	*Sam*, 2	格式化的索引項
`\index{Lin@\textbf{Lin}}`	**Lin**, 7	同上
`\index{Jenny|textbf}`	Jenny, **3**	格式化的頁碼
`\index{Joe|textit}`	Joe, *5*	同上
`\index{ecole@\'ecole}`	école, 4	重音標記

的擴展名後綴 (`.idx`)。這個 `.idx` 文件需要用 `makeindex` 程序來處理。

`makeindex` *filename*

`makeindex` 程序生成一個與源文件同名的序列化索引文件，這個文件使用 `.ind` 爲擴展名。當再次用 LaTeX 處理源文件時，這個序列化索引文件將被包含到源文件中

`\printindex`

命令出現的位置。

LaTeX 2_ε 附帶的宏包 showidx 可以在正文的左邊列印出索引項。這個功能在校對文檔和索引項時十分有用。

請注意不謹慎地使用 `\index` 命令，將會影響文檔頁版面佈局。

```
My Word \index{Word}. As opposed
to Word\index{Word}. Note the
position of the full stop.
```

My Word . As opposed to Word. Note the position of the full stop.

4.4 定製頁眉和頁腳

Piet van Oostrum 編寫的 fancyhdr 宏包[6]，提供了一些簡單的命令使得我們可以定製文檔的頁眉和頁腳。看一眼本頁的頂部，你就能發現這個宏包的用處。

定製頁眉和頁腳時一件棘手的事情就是得到每個頁面所屬的章節名稱。LaTeX 通過兩個步驟來完成這個任務。在定義頁眉和頁腳時，你可以使用 `\rightmark` 命令來代表當前的 section 名，使用 `\leftmark` 來代表當前的 chapter 名。這兩個命令的值將在處理 chapter 或者 section 命令時被重新賦值。

爲了獲得最大的靈活性，`\chapter` 等命令並不對 `\rightmark` 和 `\leftmark` 直接進行重定義，而是間接地通過調用 `\chaptermark`、 `\sectionmark` 或者 `\subsectionmark` 來重新定義 `\rightmark` 和 `\leftmark`。

因此，只需要重新定義 `\chaptermark` 命令， 就能修改頁眉上顯示的章名。

[6]可以在 `macros/latex/contrib/supported/fancyhdr` 得到。

```
\documentclass{book}
\usepackage{fancyhdr}
\pagestyle{fancy}
 with this we ensure that the chapter and section
 headings are in lowercase.
\renewcommand{\chaptermark}[1]{%
        \markboth{#1}{}}
\renewcommand{\sectionmark}[1]{%
        \markright{\thesection\ #1}}
\fancyhf{}  % delete current header and footer
\fancyhead[LE,RO]{\bfseries\thepage}
\fancyhead[LO]{\bfseries\rightmark}
\fancyhead[RE]{\bfseries\leftmark}
\renewcommand{\headrulewidth}{0.5pt}
\renewcommand{\footrulewidth}{0pt}
\addtolength{\headheight}{0.5pt} % space for the rule
\fancypagestyle{plain}{%
   \fancyhead{} % get rid of headers on plain pages
   \renewcommand{\headrulewidth}{0pt} % and the line
}
```

圖 4.1 – fancyhdr 設置實例。

圖 4.1 顯示了如何設定 fancyhdr 來得到和本文相似的頁眉。無論如何我還是建議你先閱讀一下 fancyhdr 宏包所帶的文檔。

4.5 Verbatim 宏包

在本文的前面部分你已經知道了 `verbatim` 環境。在這一節中，你將學會使用 verbatim 宏包。 verbatim 宏包重新實現了 `verbatim` 環境，並解決了原來的 `verbatim` 環境的一些限制。這本身並沒有什麼特別的，但 verbatim 宏包還實現了一些新增的功能，這才是我在這裡提到這個宏包的原因。 verbatim 宏包提供了

`\verbatiminput{`*filename*`}`

命令，這個命令允許你把一個 ASCII 碼的文本文件包含到你的文檔中來，就好像它們是在 `verbatim` 環境中一樣。

verbatim 宏包是 'tools' 宏包集的一部分，大多數的系統中都預裝了這個宏包集。如果你想更多地瞭解這個宏包，可以閱讀 [10]。

4.6 安裝額外的宏包

大多數的 LaTeX 安裝都帶有大量預裝的樣式宏包，但更多的宏包可以在網上得到。在互聯網尋找樣式宏包的一個主要的地方就是 CTAN (http://www.ctan.org/)。

各種宏包的源文件，例如 geometry， hyphenat 等等，一般來說都包含兩個文件：一個擴展名爲 `.ins`，另一個擴展名爲 `.dtx`。此外，通常會有一個 `readme.txt` 對宏包進行簡要的說明。你應該先閱讀這個文件。

無論如何，一旦你得到了宏包的源文件，你還要對它們進行處理使得 (a) 你的 TEX 系統知道這個新的宏包， (b) 生成說明文檔。下面是第一部分的步驟：

1. 對 `.ins` 文件運行 LATEX 命令。這將會產生一個 `.sty` 文件。
2. 把 `.sty` 文件移到 LATEX 系統能找到的地方。一般是在 `. . . /localtexmf/tex/latex` 子目錄下（ Windows 或者 OS/2 用戶應該改變斜線爲反斜線 ）。
3. 刷新系統的文件名資料庫。具體的命令取決於你使用的 LATEX 系統：
   teTeX, fpTeX – `texhash`; web2c – `maktexlsr`; MikTeX – `initexmf -update-fndb`
   或者使用圖形界面。

現在你可以從 `.dtx` 文件生成說明文檔：

1. 對 `.dtx` 文件運行 LATEX 命令。這會生成一個 `.dvi` 文件。注意你可能需要多次運行 LATEX 命令來正確處理交叉引用。
2. 檢查一下 LATEX 命令是否產生了 `.idx` 文件。如果沒發現這個文件，你就可以執行第 5 步了。
3. 爲了生成索引，敲入命令：
   `makeindex -s gind.ist` *name*
   （ 這裡 *name* 表示不帶擴展名的主文件名 ）。
4. 再次對 `.dtx` 文件運行 LATEX 命令。
5. 最後一步但不是必需的，生成 `.ps` 文件或者 `.pdf` 文件以方便閱讀。

有時你會看見生成了一個 `.glo` (glossary) 文件。在第 4 步和第 5 步之間運行下面的命令：

`makeindex -s gglo.ist -o` *name*`.gls` *name*`.glo`

確認在執行第 5 步前最後對 `.dtx` 文件運行一遍 LATEX 命令。

4.7 使用 pdfLATEX

By Daniel Flipo <Daniel.Flipo@univ-lille1.fr>

PDF 是一種超文本文檔 (hypertext) 格式。類似於網頁，文檔中的某些詞可以被超超連結標記。它們鏈接到這個文檔的另一個地方，甚至是另外一個文檔。如果你點擊這樣一個超超連結，你將轉調到超連結的目的地。這意味著在 LATEX 格式的文檔中所有 `\ref` 和 `\pageref` 出現的位置都變成超超連結。此外，目錄、索引和其它類似的結構變成超超連結的集合。

現在大多數的網頁都是用 HTML **超文本標記語言**編寫。在寫科技文檔的時候，這種格式有兩個嚴重的缺陷：

1. 數學公式在 HTML 文檔中通常都不被支持。儘管對此有一個標準，但是大多是現在使用的瀏覽器都不支持，或者缺少需要的字體。
2. 列印 HTML 文檔是比較容易的，但列印的結果會因系統平台和瀏覽器的不同而出現差異。這結果與我們在 LATEX 世界中期待的高質量相差十萬八千里。

有許多將 LaTeX 轉爲 HTML 的嘗試。其中一些可以說是相當成功的，它們能將一個標準的 LaTeX 源文件生成一個合格的網頁。但是爲了達到目的，需要去掉一些支持。當你開始使用 LaTeX 的複雜功能和擴展宏包時，那些嘗試就行不通了。若希望即使是發不到網上也保留文檔的高質量，作者們就要使用 PDF **（便攜式文檔格式）**，它保留了文檔的版式和允許超超連結導航。現在大多數瀏覽器只要帶上相應的插件都可以直接顯示 PDF 文檔。

儘管幾乎所有的操作系統都有 DVI 和 PS 格式的閱讀器，但你會發現人們更多地使用 Acrobat Reader 和 Xpdf 來閱讀 PDF 文檔。因而提供 PDF 格式的文檔將使得潛在的讀者更容易閱讀。

4.7.1 發佈到網上的 PDF 文檔

有了 Hàn Th´ê Thành 開發的 pdfTeX 程序，使用 LaTeX源文件來創建 PDF 文件將變得非常簡單。一般的 TeX 生成 DVI，而 pdfTeX 生成 PDF。還有 pdfLaTeX 程序將 LaTeX 源文件生成 PDF。

現在大多數的 TeX 發行版本都自動集成安裝了 pdfTeX 和 pdfLaTeX，例如： teTeX， fpTeX， MikTeX， TeXLive 和 CMacTeX。

爲了生成 PDF 而不是 DVI，只需要將命令 `latex file.tex` 改成 `pdflatex file.tex`。在不是使用命令行的 LaTeX 系統中，你可以在 TeX 控制中心找到一個專門的按鈕。

在 LaTeX 中，你可以通過 documentclass 的參數選項來定義頁面的大小，例如：`a4paper` 或 `letterpaper`。這些也對 pdfLaTeX 有效，但是首先要讓 pdfTeX 知道這種頁面的實際大小來控制 PDF 文件的頁面大小。若你使用 hyperref 宏包（參見第 63 頁），頁面的大小將自動調整。否則，你需要手動的將下面兩行加入到文檔的導言區：

```
\pdfpagewidth=\paperwidth
\pdfpageheight=\paperheight
```

接下來的一節將深入介紹 LaTeX 和 pdfLaTeX 之間的不同。主要的不同有三個方面：採用的字體，包含圖像的格式和超超連結的手動設定。

4.7.2 字體

pdfLaTeX 能處理所有的字體（PK 點陣、TrueType、 PostScript type 1 ⋯⋯），但是作爲 LaTeX 通常的字體格式， PK 點陣字體在 Acrobat Reader 下的顯示效果非常差。爲了獲得文檔更高的顯示效果，最好使用 PostScript Type 1 字體。高級的 TeX 安裝程序會自動設置好，你最好試一下，如果它正常運作，你就可以跳過這一節。

PostScript Type 1 的 Computer Modern 和 AMSFonts 字體由 Blue Sky Research 和 Y&Y, Inc. 製作，後來版權屬於美國數學學會 (AMS)。這些字體在 1997 年被開放，並且出現在大多數 TeX 發行版中。

然而，如果你使用 LaTeX 來創建非英文的文檔，你可能用到 EC， LH 或 CB 字體（關於 `OT1` 字體的討論可參見第 20 頁）。Vladimir Volovich 創建了 cm-super 字體包，包含全部 EC/TC、EC Concrete、EC Bright 和 LH 字體集。在 `CTAN:/fonts/ps-type1/cm-super` 可以獲得，也被集成進了 TeXLive7 和 MikTeX。由 Apostolos Syropoulos 創建類似 type 1 CB 的希臘字體可在 `CTAN:/tex-archive/fonts/greek/cb` 獲得。不幸的是，這些字體集跟 Blue Sky/Y&Y 的 Type 1 CM 字體的印刷質量不一樣。LaTeX 會自動提示，而且文檔在螢幕的顯示效果也不如 Blue Sky/Y&Y type 1 CM 字體那麼整潔。在高分辨率的輸出設備下，它們生成的效果跟原來的 EC/LH/CB 點陣字體一樣。

如果你使用一種拉丁語系的語文來創建文檔，你有其它一些選擇。

- 使用 aeguill 宏包，也叫 *Almost European Computer Modern with Guillemets*。只需在導言區添加一行 `\usepackage{aeguill}`，來啓用 AE 虛擬字體替代 EC 字體。
- 或者使用 mltex 宏包，但是它只在 pdfTEX 設置了 mltex 選項時有效。

AE 虛擬字體集，像 MlTEX 系統，在 CM 字體的字符基礎上創建全部缺失的字母並按 EC 順序重新排列，就使得 TEX 處理它的時候認爲它有一個全部 256 個字符的字體集。這樣就可使用大部分系統中優質的 type 1 格式的 cm 字體。這套字體現在爲 T1 編碼，在拉丁語系的歐洲語文中能很好的運作。唯一的不足就是創建的 AE 字符不支持 Acrobat Reader 的查找功能，因此你不能在 PDF 文檔中搜索那些帶重音符號的單詞。

對於俄文，一個類似的解決辦法是使用 C1 虛擬字體，這可以在 `ftp://ftp.vsu.ru/pub/tex/font` 上獲得。這套字體集成了標準的 Bluesky CM type 1 字體和 Paradissa 與 BaKoMa 的 CMCYR type 1 字體，這些都可以在 CTAN 上找到。由於 Paradissa 字體只包含俄文字母，C1 字體裡就沒有其他西里爾字符了。

另一種解決辦法是使用其它 PostScript type 1 字體。事實上，其中一些字體被包含在 Acrobat Reader 的相應語言版本中。由於這些字體有不同的字符大小，頁面上的文本格式將會改變。一般地，這類字體佔得空間要比 CM 字體大，因爲 CM 字體是一種非常節省空間的字體。而且文檔可視的一致性也被破壞了，因爲 Times、 Helvetica 和 Courier 字體（這些是 Acrobat 裡基本的可替換字體）沒有被設計來在單個文檔中和平共處。

爲了達到上述目的，有兩套字體已經做好：pxfonts，它基於作爲正文字體的 *Palatino*；另外就是基於 *Times* 的 txfonts 宏包。只需要在文檔的導言區加入下列幾行就可以使用這些字體：

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

註：當你編譯的時候，在.log 文件中出現下面的信息：

```
Warning: pdftex (file eurmo10): Font eur... not found
```

這意味著文檔使用的某些字體沒有被找到。你必須解決這些問題，否則輸出的 PDF 文件可能**不會顯示缺失字符的頁面**。

上面討論了如此多的字體問題，特別是缺乏與 type 1 格式的 CM 字體同樣高質量的 EC 字體一直困擾大家。最近一套新的被稱爲 Latin Modern (LM) 的高質量字體已開發完成。這使得前面的擔心都是多餘的。如果你安裝了最新的 TEX，你可能已經安裝好這套字體，需要做的只是在你的文檔的導言區添加

```
\usepackage{lmodern}
\usepackage[T1]{fontenc}
\usepackage{textcomp}
```

就可以創建支持所有拉丁字符的優質 PDF 輸出文件。

4.7.3 使用圖形

通過 graphicx 宏包（參見第 55 頁），你可以在文檔中插入圖形。使用 `pdftex` 作爲 *driver* 的選項，這個宏包也可用於 pdfLATEX：

```
\usepackage[pdftex]{color,graphicx}
```

上面這個例子中，我還包含了 color 宏包，因此網頁上彩色的文檔會看起來更自然一些。

好消息到此爲止。壞消息就是 Encapsulated PostScript 格式的圖形並不被 PdfLaTeX 所支持。如果你不在命令 `\includegraphics` 聲明加載的文件的擴展名，宏包 graphicx 將依賴於 *driver* 選項的設置自動尋找一個合適的文件。對於 `pdftex` ，它支持的格式有 `.png`，`.pdf`， `.jpg` 和 `.mps` (`MetaPost`)，但不支持 `.eps`。

一個解決這個問題的簡單方法是通過在很多系統中可找到的 `epstopdf` 工具將你的 EPS 文件轉化爲 PDF 格式。對於矢量圖（畫），這時一個很好的解決辦法，但對於位圖（相片、掃瞄圖），這並不是很理想，因爲 PDF 格式本來就支持包含 PNG 和 JPEG 圖像。PNG 格式適合螢幕截圖和其它一些只有較少顏色的圖像。JPEG 是一種非常適合於相片的格式，而且佔用空間少。

可能對於畫一些特殊的幾何圖形這也不是很理想，幸好可以通過一些專門的命令語言來畫圖形，例如 `MetaPost`，它可以在大多數的 TeX 發行版本中找到，並且也包含它的擴展手册。

4.7.4 超超連結

hyperref 宏包將你的文檔中的所有內部引用變成超超連結。爲此，一些特殊的設置是必須的，保證 `\usepackage[pdftex]{hyperref}` 是你的文檔導言區的**最**後一行命令。

有很多選項用於定義 hyperref 宏包的效果:

- 或者在 `\usepackage[pdftex]{hyperref}` 的 pdftex 選項後用逗號隔開列出；
- 或者使用單獨一行的命令 `\hypersetup{`*options*`}`。

`pdftex` 是唯一必須的選項，其他參數用來改變 hyperref 的默認效果[7]。下面的列表中默認值被寫成 upright 字體。

`bookmarks (=true,false)` 顯示或隱藏書籤欄

`unicode (=false,true)` 允許在 Acrobat 書籤欄使用非拉丁語系的字符

`pdftoolbar (=true,false)` 顯示或隱藏 Acrobat 的工具欄

`pdfmenubar (=true,false)` 顯示或隱藏 Acrobat 的菜單欄

`pdffitwindow (=true,false)` 調整顯示 PDF 的初始化放大倍率

`pdftitle (=`{`text`}`)`** 定義 Acrobat 顯示的文檔信息 (`Document Info`)

`pdfauthor (=`{`text`}`)`** PDF 文件作者名字

`pdfnewwindow (=true,false)` 定義當超超連結到另一個文檔時，是否打開一個新的窗口

`colorlinks (=false,true)` 超連結有一個彩色框架 (`false`) 或者鏈接文本的顏色設置 (`true`)。鏈接的顏色通過下面的參數來控制（默認的顏色已列出）

[7]值得注意的是 hyperref 宏包並不只是在 pdfTeX 下可用。它也可以用於將 PDF 特殊信息嵌入到由通常 LaTeX 生成的 DVI 文件中。然後通過 `dvips` 轉化爲 PS 格式，最後用 Adobe Distiller 將 PS 格式轉成 PDF。

`linkcolor (=red)` 內部超連結的顏色 (sections, pages, etc.),

`citecolor (=green)` 引用超連結的顏色 (bibliography)

`filecolor (=magenta)` 文件超連結的顏色

`urlcolor (=cyan)` URL 超連結的顏色 (mail, web)

如果你覺得這些默認值合適，就直接使用

```
\usepackage[pdftex]{hyperref}
```

爲了使書籤列表打開和超連結爲彩色（`=true` 不需要寫出來）：

```
\usepackage[pdftex,bookmarks,colorlinks]{hyperref}
```

當創建 PDF 文檔是用來列印的，使用彩色的超連結並不是一件好事，因爲彩色的鏈接將會被列印成灰色，從而難以閱讀。你可以設置不列印彩色的框架：

```
\usepackage{hyperref}
\hypersetup{colorlinks=false}
```

或者將超超連結變成黑色：

```
\usepackage{hyperref}
\hypersetup{colorlinks,%
            citecolor=black,%
            filecolor=black,%
            linkcolor=black,%
            urlcolor=black,%
            pdftex}
```

當你只想提供 PDF 文件的文檔信息 (`Document Info`)：

```
\usepackage[pdfauthor={Pierre Desproges},%
            pdftitle={Des femmes qui tombent},%
            pdftex]{hyperref}
```

除了自動的交叉引用的超超連結，通過下面的命令可以嵌入明確的鏈接：

`\href{`*url*`}{`*text*`}`

代碼

```
The \href{http://www.ctan.org}{CTAN} website.
```

生成的效果爲 “CTAN”；單擊詞 “CTAN” 將把你帶到 CTAN 網站。

若超連結的目的地不是一個 URL ，而是一個本地文件，你可以使用 `\href` 命令：

```
The complete document is \href{manual.pdf}{here}
```

生成的效果爲 “The complete document is here”。單擊 “here” 將打開文件 `manual.pdf`（文件名跟依賴於當前文檔的位置）。

若文章的作者希望讀者可以容易地發郵件給她，只需要在文檔的標題頁的 `\author` 命令中使用命令 `\href`：

```
\author{Mary Oetiker $<$\href{mailto:mary@oetiker.ch}%
       {mary@oetiker.ch}$>$
```

注意到這個超連結不僅顯示在鏈接中還顯示在頁面上。下面的鏈接
`\href{mailto:mary@oetiker.ch}{Mary Oetiker}`
在 Acrobat 中可正常使用，但頁面被列印時，郵箱地址將不可見。

4.7.5 超連結的問題

當一個計數器被重新初始化後可能出現下面的信息：

```
! pdfTeX warning (ext4): destination with the same
  identifier (name{page.1}) has been already used,
  duplicate ignored
```

例如，在 book 類的文檔中使用命令 \mainmatter 就可能出現上面的警告。它在書的第一章重設頁碼計數器爲 1，但是書的序言部分也有頁碼 1，從而超連結到「頁碼 1」不再是唯一的，故出現了 "duplicate has been ignored."

一個解決的辦法是將 plainpages=false 加入到 hyperref 選項中。不幸的是這樣只對頁碼計數器有效。或者冒險使用 hypertexnames=false，但是這會使得索引中的頁碼超連結失效。

4.7.6 書籤的問題

書籤中的文本有時並不會像你想像中的那樣顯示，因爲書籤僅僅是「文本」，其中可使用的字符要比正常 LaTeX 文件的少得多。Hyperref 經常遇到這類問題而出現下面的警告：

```
Package hyperref Warning:
Token not allowed in a PDFDocEncoded string:
```

現在你可以通過爲書籤提供一個文本字符串來解決這個問題，用

\texorpdfstring{*TEX text*}{*Bookmark Text*}

來替換不正常顯示的文本。

數學表達式也是這類問題的典型代表：

```
\section{\texorpdfstring{$E=mc^2$}%
        {E=mc^2}}
```

而通常在書籤中 \section{$E=mc^2$} 顯示爲「E=mc2」。

顏色的改變也在書籤顯示中出問題：

```
\section{\textcolor{red}{Red !}}
```

顯示的結果爲「redRed!」。命令 \textcolor 被忽略。而它的參數 (red) 被輸出。

如果你使用

```
\section{\texorpdfstring{\textcolor{red}{Red !}}{Red\ !}}
```

結果將更美觀。

如果你寫 unicode （統一字符編碼）類型的文檔，就需添加宏包 hyperref 的選項 unicode，這樣你就可以在書籤中使用 unicode 字符。然後使用 \texorpdfstring 命令將讓你有較大範圍的字符供選擇。

LaTeX 和 pdfLaTeX 的源文件的兼容性

理想的話，你的文檔用 LaTeX 和 pdfLaTeX 編譯的效果應該一致，這方面的問題主要是包含的圖像。一個簡單的解決辦法是在命令 `\includegraphics` 中不設置加載的圖像的擴展名，它們會自動在當前目錄尋找一個格式適合的文件，你需要做的是創建相應格式的圖像文件。LaTeX 會尋找 `.eps` 格式，而 pdfLaTeX 會嘗試包含一個擴展名爲 `.png`， `.pdf`，`.jpg` 或 `.mps` （按這個順序）的文件。

若你想在 PDF 格式的文件中使用不同的代碼，只需要簡單地在導言區加入宏包 ifpdf [8]。但前提是你已經安裝了這個宏包，如果沒有安裝而你又使用 MiKTeX 的話，它將在你第一次使用的時候自動安裝。這個宏包定義了一個特殊的命令 `\ifpdf`，利用它你可很容易編寫條件代碼。例如，考慮到列印費用，用 PostScript 格式僅使用黑白色，但在線查看的 PDF 格式將是彩色的。

```
\RequirePackage{ifpdf} % running on pdfTeX?
\ifpdf
  \documentclass[a4paper,12pt,pdftex]{book}
\else
  \documentclass[a4paper,12pt,dvips]{book}
\fi

\ifpdf
  \usepackage{lmodern}
\fi
\usepackage[bookmarks, % add hyperlinks
            colorlinks,
            plainpages=false]{hyperref}
\usepackage[T1]{fontenc}
\usepackage[latin1]{inputenc}
\usepackage[english]{babel}
\usepackage{graphicx}
...
```

在上面的例子中，在非 PDF 格式中我也包含了 hyperref 宏包，這樣 `\href` 命令在所有情形下都有效，這也使得我不用在每個情況下都使用條件聲明。

注意到當前的 TeX 發行版本（例如 TeXLive），通常的 TeX 會根據文檔類型的設置自動選擇輸出 PDF 還是 DVI。如果你使用上面的代碼，你仍然可以使用 `pdflatex` 命令來得到 PDF 格式的輸出或使用 `latex` 得到 DVI 格式。

4.8 創建演示文稿

By Daniel Flipo <Daniel.Flipo@univ-lille1.fr>

你可以將你的科學工作成果通過黑板、透明片或者在你的筆記本電腦上直接使用演示文稿軟件呈現。

pdfLaTeX 和 beamer 文檔類允許你創建 PDF 格式的演示文稿，結果跟你用一天時間製作的 PowerPoint 看上去差不多，但更便攜因爲 Acrobat Reader 支持更多的系統平台。

beamer 文檔類使用帶參數的宏包 graphicx、 color 和 hyperref 來適應螢幕閱讀的演示文稿。

[8] 如果你想知道爲什麼要使用這個宏包，可以參見 TeX FAQ 的這個欄目 http://www.tex.ac.uk/cgi-bin/texfaq2html?label=ifpdf。

```
\documentclass[10pt]{beamer}
\mode<beamer>{%
  \usetheme[hideothersubsections,
            right,width=22mm]{Goettingen}
}

\title{Simple Presentation}
\author[D. Flipo]{Daniel Flipo}
\institute{U.S.T.L. \& GUTenberg}
\titlegraphic{\includegraphics[width=20mm]{USTL}}
\date{2005}

\begin{document}

\begin{frame}<handout:0>
  \titlepage
\end{frame}

\section{一個例子}

\begin{frame}
  \frametitle{Things to do on a Sunday Afternoon}
  \begin{block}{One could \ldots}
    \begin{itemize}
      \item walk the dog\dots \pause
      \item read a book\pause
      \item confuse a cat\pause
    \end{itemize}
  \end{block}
  and many other things
\end{frame}
\end{document}
```

圖 4.2 – beamer 文檔類的範例。

當你用 PDFLaTeX 編譯圖 4.2 中的代碼時，你將得到一個 PDF 文件，第一頁爲標題頁，第二頁有幾個欄目，但當你單擊你的演示文檔時，一次顯示一條欄目。

beamer 類創建 PDF 文件的一個優點是直接生成可用的文檔，而不像 prosper 需要先通過一個 PostScript 步驟，也不像 ppower4 宏包需要一個後加工處理才能生成演示文檔。

用 beamer 類，你可以用一個源文件生成幾種版本。可以在源文件的中括弧中加入特定的選項來生成不同的版本。有下面幾種版式：

beamer PDF 螢幕閱讀版本；

trans 幻燈片版本；

handout PDF 講義版本。

默認的版本爲 beamer，你可以通過設置不同的全局選項來修改，例如：用 `\documentclass[10pt,handout]{beamer}` 來生成講義版本。

演示文稿外觀依賴於你選擇的主題。你可以選擇 beamer 類自帶的一個主題，也可以自己定義一個新的主題。詳情請參見 beamer 類的幫助文檔 `beameruserguide.pdf`。

讓我們再來仔細分析圖 4.2 中的代碼。

對於螢幕閱讀版本的演示文稿 `\mode<beamer>`，我們選擇了 *Goettingen* 主題，它將目錄合成到導航面板。通過選項控製麵板的大小（這個例子採用 22 mm），和確定面板的位置（正文右側）。選項 *hideothersubsections* 顯示章節的標題，但只顯示當前章節的子節標題。對於 `\mode<trans>` 和 `\mode<handout>` 的設置也是一樣的，它們將出現在它們標準的版面上。

命令 `\title{}`，`\author{}`，`\institute{}` 和 `\titlegraphic{}` 定義標題頁的內容。`\title[]{}` 和 `\author[]{}` 的選項允許你定義顯示在 *Goettingen* 主題的面板上的標題和作者名。

面板中的標題和子標題由 `frame` 環境外面的命令 `\section{}` 和 `\subsection{}` 來創建。

螢幕底部的一些微型導航圖標也可以讓你瀏覽整個文檔。它們的出現不依賴你選擇的主題。

每張幻燈片或每版螢幕的內容放在 `frame` 環境中。利用尖括弧（< 和 >）裡面的選項，用演示文檔的一個版式來定義一個特殊的幀。在這個例子中，第一頁不會由於參量 `<handout:0>` 而顯示爲講義模式。

除了幻燈片的標題頁，強烈建議通過命令 `\frametitle{}` 來重新設置每一張幻燈片的標題。如果需要，使用 `block` 環境可以來定義子標題，在這個例子中也可體現出來。注意到章節命令 `\section{}` 和 `\subsection{}` 不在幻燈片上產生輸出結果。

列表環境中的命令 `\pause` 允許你一個接一個地顯示列表欄目的內容。命令：`\only`、`\uncover`、 `\alt` 和 `\temporal`，可以讓你獲得其他的一些演示效果。很多情況下，你可以通過尖括弧中的內容來定製演示效果。

無論如何，建議你閱讀 beamer 類的文檔 `beameruserguide.pdf` 來獲得一個全面的瞭解。這個宏包正在活躍地開發中，去它們的網站 http://latex-beamer.sourceforge.net/ 可獲取最新的信息。

Chapter 5

數學圖形

大部分人使用 LaTeX排版文本。但是因爲這種不面向內容和結構的寫作方法太方便了， LaTeX還提供了從文本描述生成圖形輸出的一種有局限性的方法。此外，大量的 LaTeX擴展被開發出來以克服這些限制。在本節中，我們將學習其中的一些。

5.1 概述

`picture` 環境可以在 LaTeX 裡直接設計圖形。詳細的介紹請參考 *LaTeX Manual* [1]。一方面，這種方法有嚴重的局限性，比如線段的斜率和圓的半徑只能在一個很小的範圍內取值。另一方面， LaTeX 2_ε的 `picture` 環境提供了 `\qbezier` 命令， "q" 表示 "quadratic"。許多常用的曲線如圓、橢圓、或者懸鏈線都可以用二次 Bézier 曲線得到令人滿意的近似，雖然這可能需要一些辛苦的數學準備。另外，如果有一種編程語言如 Java 能用來生成 LaTeX源文檔的 `\qbezier` 模塊， `picture` 環境會更強大。

雖然直接在 LaTeX裡設計圖形的方法有嚴重的局限性而且通常比較繁瑣，但它還是很有用的。這份文檔就是用它才變得體積很小，不需要插入額外的圖片。

一些宏包，如 epic 和 eepic（ *The LaTeX Companion* [3]裡有介紹），或者 pstricks 可以排除 `picture` 環境的局限，並大大地增強了 LaTeX的圖形功能。

跟前兩個宏包只是加強了 `picture` 環境不同，pstricks 宏包有自己的繪圖環境， `pspicture`。 pstricks 的強大之處在於它廣泛應用了 PostScript。另外，許多宏包可以用來處理專門的問題。其一是 XY-pic，本章最後會講到它。 *The LaTeX Graphics Companion* [4]（勿與 *The LaTeX Companion* [3]混淆）裡詳細介紹了大量的宏包.

LaTeX最強大的圖形工具可能是 `MetaPost`， Donald E. Knuth 編寫的 `METAFONT` 的孿生兄弟。 `MetaPost` 使用非常強大的數學編程語言： `METAFONT`。與 `METAFONT` 生成點陣圖片不同，`MetaPost` 生成的是封裝的 PostScript 文件，可以導入 LaTeX中。其介紹可以看 *A User's Manual for MetaPost* [15]，或者 [17]。

關於 LaTeX 和 TeX 圖形（以及字體）支持方法的詳細討論請參考 *TeX Unbound* [16]。

5.2 picture 環境

By Urs Oswald <osurs@bluewin.ch>

5.2.1 基本命令

一個 picture 環境[1]可以用下面兩個命令中的一個來創建

`\begin{picture}(`x, y`)`...`\end{picture}`

或者

`\begin{picture}(`x, y`)(`x_0, y_0`)`...`\end{picture}`

數字 x, y, x_0, y_0 是相對於 \unitlength 而言的，任何時候 (除了在 picture 環境之內以外) ，都可以使用命令如

`\setlength{\unitlength}{1.2cm}`

來改變。\unitlength 的默認值是 1 pt。第一個數對， (x, y)，在文檔中爲圖形保留一個矩形的區域。可選的第二個數對， (x_0, y_0)，爲矩形左下角指派任意的坐標。

大多數的繪圖命令是下面兩種格式之一

`\put(`x, y`){`*object*`}`

或者

`\multiput(`x, y`)(`$\Delta x, \Delta y$`){`n`}{`*object*`}`

Bézier 曲線是一個例外。它們需要用命令

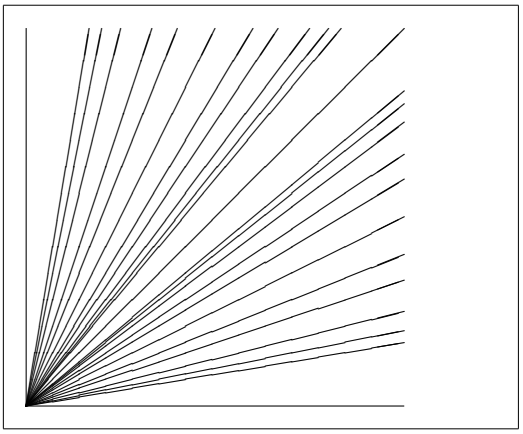
`\qbezier(`x_1, y_1`)(`x_2, y_2`)(`x_3, y_3`)`

來畫。

[1]信不信由你，picture 環境僅需標準的 LaTeX 2_ε，「開箱即用」，無需載入宏包。

5.2.2 線段

```
\setlength{\unitlength}{5cm}
\begin{picture}(1,1)
  \put(0,0){\line(0,1){1}}
  \put(0,0){\line(1,0){1}}
  \put(0,0){\line(1,1){1}}
  \put(0,0){\line(1,2){.5}}
  \put(0,0){\line(1,3){.3333}}
  \put(0,0){\line(1,4){.25}}
  \put(0,0){\line(1,5){.2}}
  \put(0,0){\line(1,6){.1667}}
  \put(0,0){\line(2,1){1}}
  \put(0,0){\line(2,3){.6667}}
  \put(0,0){\line(2,5){.4}}
  \put(0,0){\line(3,1){1}}
  \put(0,0){\line(3,2){1}}
  \put(0,0){\line(3,4){.75}}
  \put(0,0){\line(3,5){.6}}
  \put(0,0){\line(4,1){1}}
  \put(0,0){\line(4,3){1}}
  \put(0,0){\line(4,5){.8}}
  \put(0,0){\line(5,1){1}}
  \put(0,0){\line(5,2){1}}
  \put(0,0){\line(5,3){1}}
  \put(0,0){\line(5,4){1}}
  \put(0,0){\line(5,6){.8333}}
  \put(0,0){\line(6,1){1}}
  \put(0,0){\line(6,5){1}}
\end{picture}
```



線段用命令

`\put(`x, y`){\line(`x_1, y_1`){`$length$`}}`

來畫。命令 `\line` 有兩個參量：

1. 一個方向向量，

2. 一個長度。

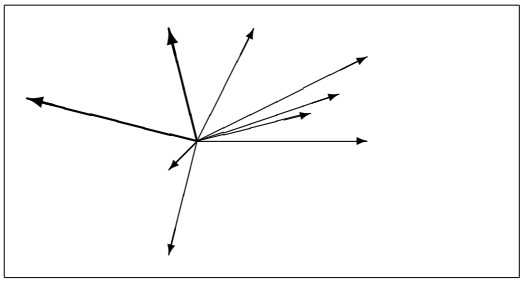
方向向量需由以下整數構成

$$-6,\ -5,\ \ldots,\ 5,\ 6,$$

而且它們需要*互質*（除 1 以外，沒有公約數），圖形顯示了第一象限中所有 25 個可能的斜率值。長度是相對於 `\unitlength` 來說的。長度的參量當一個垂直線段時是垂直坐標，其他情況都是水平坐標。

5.2.3 箭頭

```
\setlength{\unitlength}{0.75mm}
\begin{picture}(60,40)
  \put(30,20){\vector(1,0){30}}
  \put(30,20){\vector(4,1){20}}
  \put(30,20){\vector(3,1){25}}
  \put(30,20){\vector(2,1){30}}
  \put(30,20){\vector(1,2){10}}
  \thicklines
  \put(30,20){\vector(-4,1){30}}
  \put(30,20){\vector(-1,4){5}}
  \thinlines
  \put(30,20){\vector(-1,-1){5}}
  \put(30,20){\vector(-1,-4){5}}
\end{picture}
```



畫箭頭要用命令

`\put(`x, y`){\vector(`x_1, y_1`){`$length$`}}`

箭頭的方向向量元素比線段的限制更嚴格，需由以下整數構成

$$-4,\ -3,\ \ldots,\ 3,\ 4.$$

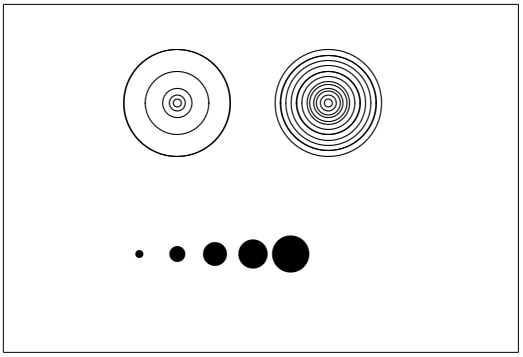
而且需要互質（除 1 以外，沒有公約數）。注意命令 `\thicklines` 對指向左上方的兩個箭頭產生的效果。

5.2.4 圓

```
\setlength{\unitlength}{1mm}
\begin{picture}(60, 40)
  \put(20,30){\circle{1}}
  \put(20,30){\circle{2}}
  \put(20,30){\circle{4}}
  \put(20,30){\circle{8}}
  \put(20,30){\circle{16}}
  \put(20,30){\circle{32}}

  \put(40,30){\circle{1}}
  \put(40,30){\circle{2}}
  \put(40,30){\circle{3}}
  \put(40,30){\circle{4}}
  \put(40,30){\circle{5}}
  \put(40,30){\circle{6}}
  \put(40,30){\circle{7}}
  \put(40,30){\circle{8}}
  \put(40,30){\circle{9}}
  \put(40,30){\circle{10}}
  \put(40,30){\circle{11}}
  \put(40,30){\circle{12}}
  \put(40,30){\circle{13}}
  \put(40,30){\circle{14}}

  \put(15,10){\circle*{1}}
  \put(20,10){\circle*{2}}
  \put(25,10){\circle*{3}}
  \put(30,10){\circle*{4}}
  \put(35,10){\circle*{5}}
\end{picture}
```



命令

`\put(`x, y`){\circle{`*diameter*`}}`

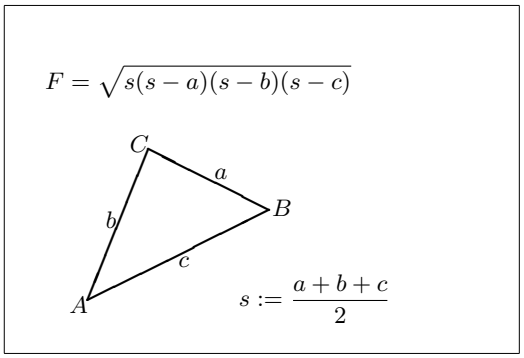
畫了一個圓心在 (x, y) 直徑（不是半徑）爲 *diameter* 的圓。 `picture` 環境只允許直徑最大是 14 mm，而且即使在這個限制之下，也不是所有的直徑都可獲得。命令 `\circle*` 生成圓盤（填充的圓形）。

跟線段的情況一樣，你可能需要其他宏包的幫助，比如 eepic 或者 pstricks。這些宏包的詳細說明請參考 *The LaTeX Graphics Companion* [4]。

`picture` 環境還有另外一個可能。如果你不怕麻煩的必要的計算（或者交給一個程序來處理），任意的圓和矩形都可以由二次 Bézier 曲線拼成。請看例子 *Graphics in LaTeX 2ε* [17]以及 Java 源文件。

5.2.5 文本與公式

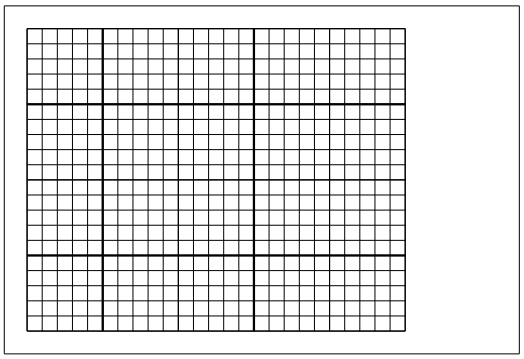
```
\setlength{\unitlength}{0.8cm}
\begin{picture}(6,5)
  \thicklines
  \put(1,0.5){\line(2,1){3}}
  \put(4,2){\line(-2,1){2}}
  \put(2,3){\line(-2,-5){1}}
  \put(0.7,0.3){$A$}
  \put(4.05,1.9){$B$}
  \put(1.7,2.95){$C$}
  \put(3.1,2.5){$a$}
  \put(1.3,1.7){$b$}
  \put(2.5,1.05){$c$}
  \put(0.3,4){$F=
    \sqrt{s(s-a)(s-b)(s-c)}$}
  \put(3.5,0.4){$\displaystyle
    s:=\frac{a+b+c}{2}$}
\end{picture}
```



如本例所示，文本與公式可以使用 \put 命令按照正常方式在 picture 環境中使用。

5.2.6 \multiput 與 \linethickness

```
\setlength{\unitlength}{2mm}
\begin{picture}(30,20)
  \linethickness{0.075mm}
  \multiput(0,0)(1,0){26}%
    {\line(0,1){20}}
  \multiput(0,0)(0,1){21}%
    {\line(1,0){25}}
  \linethickness{0.15mm}
  \multiput(0,0)(5,0){6}%
    {\line(0,1){20}}
  \multiput(0,0)(0,5){5}%
    {\line(1,0){25}}
  \linethickness{0.3mm}
  \multiput(5,0)(10,0){2}%
    {\line(0,1){20}}
  \multiput(0,5)(0,10){2}%
    {\line(1,0){25}}
\end{picture}
```



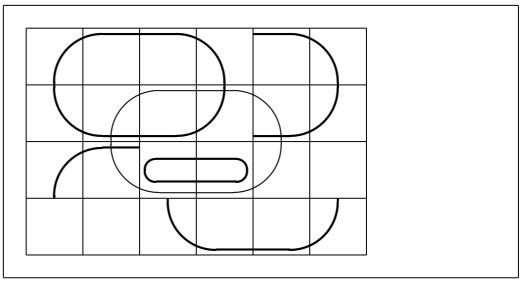
命令

\multiput$(x,y)(\Delta x,\Delta y)\{n\}\{object\}$

有 4 個參量：初始點，從一個對象到下一個的平移向量，對象的數目和要繪製的對象。命令 \linethickness 可作用於水平和垂直方向的線段，但不能作用於傾斜的線段和圓。然而，該命令可作用於二次 Bézier 曲線。

5.2.7 橢圓

```
\setlength{\unitlength}{0.75cm}
\begin{picture}(6,4)
  \linethickness{0.075mm}
  \multiput(0,0)(1,0){7}%
    {\line(0,1){4}}
  \multiput(0,0)(0,1){5}%
    {\line(1,0){6}}
  \thicklines
  \put(2,3){\oval(3,1.8)}
  \thinlines
  \put(3,2){\oval(3,1.8)}
  \thicklines
  \put(2,1){\oval(3,1.8)[tl]}
  \put(4,1){\oval(3,1.8)[b]}
  \put(4,3){\oval(3,1.8)[r]}
  \put(3,1.5){\oval(1.8,0.4)}
\end{picture}
```



命令

`\put(`x, y`){\oval(`w, h`)}`

或

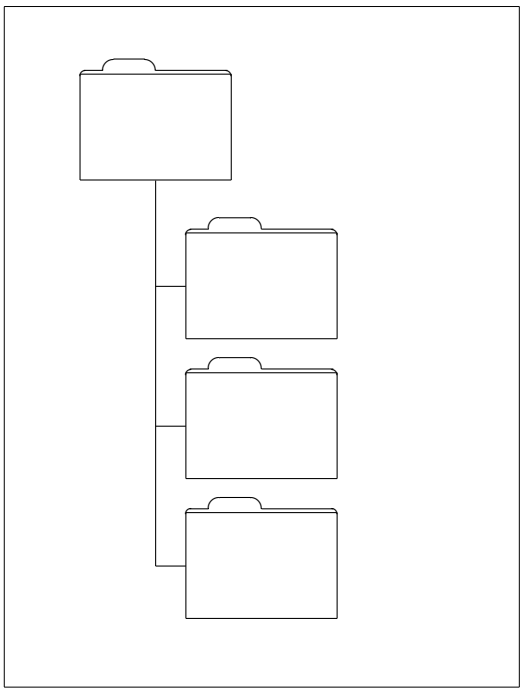
`\put(`x, y`){\oval(`w, h`)[`*position*`]}`

可以產生一個中心在 (x, y) 處、寬爲 w 高爲 h 的橢圓。如本例所示，可選參量 *position* 可以是 `b`, `t`, `l`, `r`, 分別表示僅繪製橢圓的「下部」、「上部」、「左部」和「右部」，如例所示，這些參數可以進行組合。

以下兩類命令可以控制線寬：一類爲 `\linethickness{`*length*`}`，另一類爲 `\thinlines` 與 `\thicklines`。命令 `\linethickness{`*length*`}` 僅對水平和垂直直線（及二次 Bézier 曲線）有作用， `\thinlines` 與 `\thicklines` 則可以作用於傾斜的線段、圓和橢圓。

5.2.8 重複使用預定義的圖形盒子

```
\setlength{\unitlength}{0.5mm}
\begin{picture}(120,168)
\newsavebox{\foldera}
\savebox{\foldera}
  (40,32)[bl]{% definition
  \multiput(0,0)(0,28){2}
    {\line(1,0){40}}
  \multiput(0,0)(40,0){2}
    {\line(0,1){28}}
  \put(1,28){\oval(2,2)[tl]}
  \put(1,29){\line(1,0){5}}
  \put(9,29){\oval(6,6)[tl]}
  \put(9,32){\line(1,0){8}}
  \put(17,29){\oval(6,6)[tr]}
  \put(20,29){\line(1,0){19}}
  \put(39,28){\oval(2,2)[tr]}
}
\newsavebox{\folderb}
\savebox{\folderb}
  (40,32)[l]{%          definition
  \put(0,14){\line(1,0){8}}
  \put(8,0){\usebox{\foldera}}
}
\put(34,26){\line(0,1){102}}
\put(14,128){\usebox{\foldera}}
\multiput(34,86)(0,-37){3}
  {\usebox{\folderb}}
\end{picture}
```



一個圖形盒子可以使用命令

`\newsavebox{`*name*`}`

進行**聲明**，然後使用命令

`\savebox{`*name*`}(`*width*`,`*height*`)[`*position*`]{`*content*`}`

進行**定義**，最後使用命令

`\put(`x, y`)\usebox{`*name*`}`

進行任意次數的重複**繪製**。

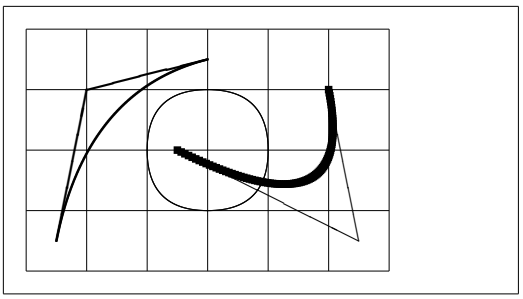
可選參數 *position* 的作用是定義圖形存放盒子的「錨點」。在本例中該參數被設置爲 `bl`，從而將錨點設置爲圖形存放盒子的左下角。其他的位置描述有 `t` 和 `r`，分別表示「上」和「右」。

參量 *name* 指明了 LaTeX 存儲槽，揭示了其命令本質（在本例中指反斜線）。圖形盒子可以嵌套：在本例中，`\foldera` 被用在了 `\folderb` 的定義中。

由於命令 `\line` 在線段長度小於大約 3 mm 的時候不能正常工作，所以必須使用命令 `\oval`。

5.2.9 二次 Bézier 曲線

```
\setlength{\unitlength}{0.8cm}
\begin{picture}(6,4)
  \linethickness{0.075mm}
  \multiput(0,0)(1,0){7}
    {\line(0,1){4}}
  \multiput(0,0)(0,1){5}
    {\line(1,0){6}}
  \thicklines
  \put(0.5,0.5){\line(1,5){0.5}}
  \put(1,3){\line(4,1){2}}
  \qbezier(0.5,0.5)(1,3)(3,3.5)
  \thinlines
  \put(2.5,2){\line(2,-1){3}}
  \put(5.5,0.5){\line(-1,5){0.5}}
  \linethickness{1mm}
  \qbezier(2.5,2)(5.5,0.5)(5,3)
  \thinlines
  \qbezier(4,2)(4,3)(3,3)
  \qbezier(3,3)(2,3)(2,2)
  \qbezier(2,2)(2,1)(3,1)
  \qbezier(3,1)(4,1)(4,2)
\end{picture}
```



如本例所示，將圓分割爲 4 條二次 Bézier 曲線的效果不能令人滿意，至少需要 8 條。該圖再一次展示了命令 `\linethickness` 對水平或垂直直線以及命令 `\thinlines` 和 `\thicklines` 對傾斜線段的影響。該例同時顯示：這兩類命令都會影響二次 Bézier 曲線，每一條命令都會覆蓋以前所有命令。

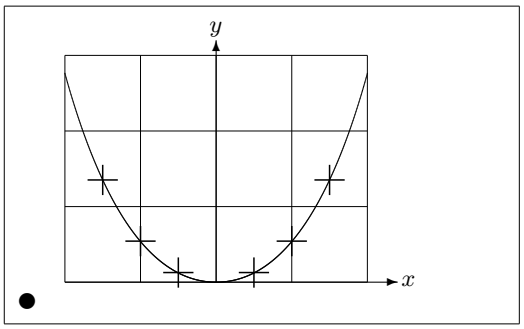
令 $P_1 = (x_1, y_1)$, $P_2 = (x_2, y_2)$ 和 m_1, m_2 分別表示一條二次 Bézier 曲線的兩個端點及其對應斜率。中間控制點 $S = (x, y)$ 則由下述方程給出

$$\begin{cases} x &= \dfrac{m_2 x_2 - m_1 x_1 - (y_2 - y_1)}{m_2 - m_1}, \\ y &= y_i + m_i(x - x_i) \qquad (i = 1, 2). \end{cases} \tag{5.1}$$

關於生成必要的 `\qbezier` 命令的 Java 程序參見 *Graphics in LaTeX 2ε* [17]。

5.2.10 懸鏈線

```
\setlength{\unitlength}{1cm}
\begin{picture}(4.3,3.6)(-2.5,-0.25)
\put(-2,0){\vector(1,0){4.4}}
\put(2.45,-.05){$x$}
\put(0,0){\vector(0,1){3.2}}
\put(0,3.35){\makebox(0,0){$y$}}
\qbezier(0.0,0.0)(1.2384,0.0)
  (2.0,2.7622)
\qbezier(0.0,0.0)(-1.2384,0.0)
  (-2.0,2.7622)
\linethickness{.075mm}
\multiput(-2,0)(1,0){5}
  {\line(0,1){3}}
\multiput(-2,0)(0,1){4}
  {\line(1,0){4}}
\linethickness{.2mm}
\put( .3,.12763){\line(1,0){.4}}
\put(.5,-.07237){\line(0,1){.4}}
\put(-.7,.12763){\line(1,0){.4}}
\put(-.5,-.07237){\line(0,1){.4}}
\put(.8,.54308){\line(1,0){.4}}
\put(1,.34308){\line(0,1){.4}}
\put(-1.2,.54308){\line(1,0){.4}}
\put(-1,.34308){\line(0,1){.4}}
\put(1.3,1.35241){\line(1,0){.4}}
\put(1.5,1.15241){\line(0,1){.4}}
\put(-1.7,1.35241){\line(1,0){.4}}
\put(-1.5,1.15241){\line(0,1){.4}}
\put(-2.5,-0.25){\circle*{0.2}}
\end{picture}
```



在本圖中，懸鏈線 $y = \cosh x - 1$ 對稱的兩半由二次 Bézier 曲線分別近似地繪成。曲線的右半部分終止於點 $(2, 2.7622)$，對應的斜率爲 $m = 3.6269$。再次使用公式 (5.1)，我們可以計算中間控制點。計算結果爲 $(1.2384, 0)$ 和 $(-1.2384, 0)$。圖中的十字爲**真**正的懸鏈線上的點。誤差小於百分之一，很難被發現。

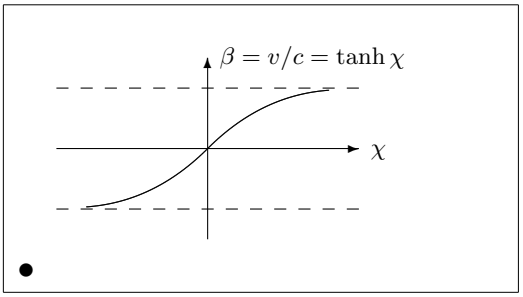
該例指出了命令 `\begin{picture}` 的可選參數的用法。該圖通過使用命令

```
\begin{picture}(4.3,3.6)(-2.5,-0.25)
```

定義了方便的「數學」坐標：左下角（由黑色圓點標出）坐標是 $(-2.5, -0.25)$。

5.2.11 坐標的相對性

```
\setlength{\unitlength}{0.8cm}
\begin{picture}(6,4)(-3,-2)
  \put(-2.5,0){\vector(1,0){5}}
  \put(2.7,-0.1){$\chi$}
  \put(0,-1.5){\vector(0,1){3}}
  \multiput(-2.5,1)(0.4,0){13}
    {\line(1,0){0.2}}
  \multiput(-2.5,-1)(0.4,0){13}
    {\line(1,0){0.2}}
  \put(0.2,1.4)
    {$\beta=v/c=\tanh\chi$}
  \qbezier(0,0)(0.8853,0.8853)
    (2,0.9640)
  \qbezier(0,0)(-0.8853,-0.8853)
    (-2,-0.9640)
  \put(-3,-2){\circle*{0.2}}
\end{picture}
```



公式 (5.1) 給出了兩條 Bézier 曲線的控制點。正向分支由 $P_1 = (0,\,0)$, $m_1 = 1$ 和 $P_2 = (2,\,\tanh 2)$, $m_2 = 1/\cosh^2 2$ 確定。與前例相同，本圖也定義了在數學上方便的坐標，左下角的坐標是 $(-3, -2)$（黑點）。

5.3 XY-pic

By Alberto Manuel Brandão Simões <albie@alfarrabio.di.uminho.pt>

xy 是繪製流程圖的專用宏包。要想使用它，只需在導言區加上：

```
\usepackage[options]{xy}
```

options 列出你需要載入的 XY-pic 的選項。這些選項基本上被用於調試這個宏包的使用。建議你使用 `all`，可以讓 LaTeX 載入 XY 的所有命令。

XY-pic 流程圖被繪製在一幅以矩陣定位的畫布上，每一個流程圖元素被放在矩陣的一個單元中：

```
\begin{displaymath}
\xymatrix{A & B \\
          C & D }
\end{displaymath}
```

$$\begin{displaymath}
\xymatrix{A & B \\
          C & D }
\end{displaymath}$$

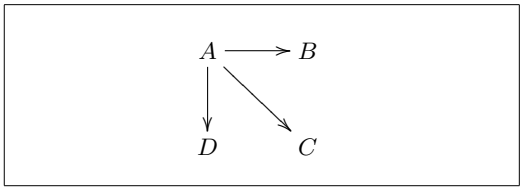
命令 `\xymatrix` 必須置於數學模式中。這裡，我們設定了一個兩行兩列的矩陣。爲了畫出流程，我們只需要使用命令 `\ar` 增加帶方向的箭頭即可。

```
\begin{displaymath}
\xymatrix{ A \ar[r] & B \ar[d] \\
           D \ar[u] & C \ar[l] }
\end{displaymath}
```

$$\begin{displaymath}
\xymatrix{ A \ar[r] & B \ar[d] \\
           D \ar[u] & C \ar[l] }
\end{displaymath}$$

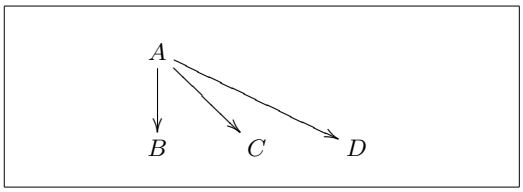
箭頭命令要放在其出發的那個單元裡。參量是箭頭的方向（u：上， d：下， r：右以及 l：左）.

```
\begin{displaymath}
\xymatrix{
  A \ar[d] \ar[dr] \ar[r] & B \\
  D                       & C }
\end{displaymath}
```



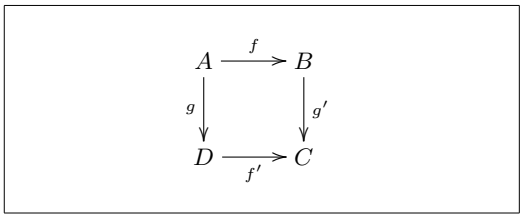
要畫對角線，可以指定不只一個方向參量。實際上，你還可以重複同一個方向來得到更大的箭頭。

```
\begin{displaymath}
\xymatrix{
 A \ar[d] \ar[dr] \ar[drr] &&\\
 B                   & C & D }
\end{displaymath}
```



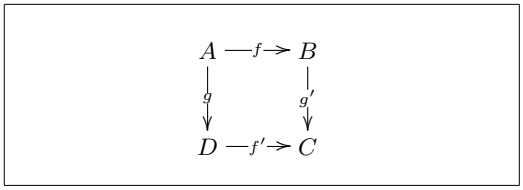
我們還可以繪製一些更有趣的流程圖，給箭頭加上標籤，只需要使用普通的上標和下標。

```
\begin{displaymath}
\xymatrix{
  A \ar[r]^f \ar[d]_g &
            B \ar[d]^{g'} \\
  D \ar[r]_{f'}       & C }
\end{displaymath}
```



如圖所示，就像數學模式裡一樣使用上下標。唯一的區別在於：上標表示放在「箭頭的上方」，下標表示放在「箭頭的下方」。把文本放到箭頭上可以用 |。

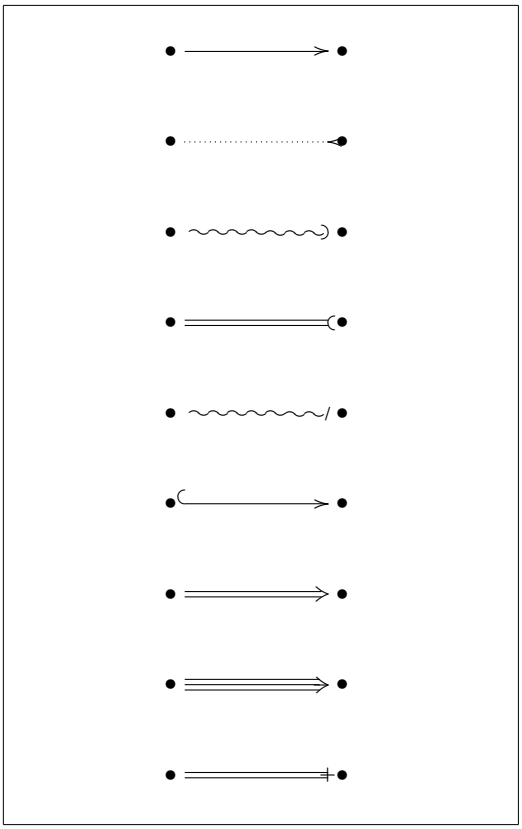
```
\begin{displaymath}
\xymatrix{
  A \ar[r]|f \ar[d]|g &
            B \ar[d]|{g'} \\
  D \ar[r]|{f'}       & C }
\end{displaymath}
```



繪製空心箭頭的命令是 \ar[...]|\hole。

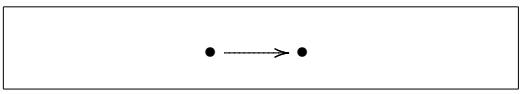
某些情況下，需要區分不同類型的箭頭。可以給它們標上標籤，或者使用不同的外觀來實現：

```
\shorthandoff{"}
\begin{displaymath}
\xymatrix{
\bullet\ar@{->}[rr] && \bullet\\
\bullet\ar@{.<}[rr] && \bullet\\
\bullet\ar@{~)}[rr] && \bullet\\
\bullet\ar@{=(}[rr] && \bullet\\
\bullet\ar@{~/}[rr] && \bullet\\
\bullet\ar@{^{(}->}[rr] &&
                        \bullet\\
\bullet\ar@2{->}[rr] && \bullet\\
\bullet\ar@3{->}[rr] && \bullet\\
\bullet\ar@{=+}[rr]  && \bullet
}
\end{displaymath}
\shorthandon{"}
```

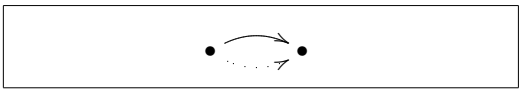


注意下面兩幅流程圖的區別：

```
\begin{displaymath}
\xymatrix{
 \bullet \ar[r]
         \ar@{.>}[r] &
 \bullet
}
\end{displaymath}
```



```
\begin{displaymath}
\xymatrix{
 \bullet \ar@/^/[r]
         \ar@/_/@{.>}[r] &
 \bullet
}
\end{displaymath}
```



兩條斜線間的修飾元素決定了曲線應該如何被畫出。 XY-pi 提供了很多辦法來改變曲線的形狀；更詳細的內容請參考 XY-pic 的文檔。

Chapter 6

定製 LaTeX

到目前爲止，運用你所學過的命令可以製作出能被絕大多數讀者接受的文檔。儘管這些文檔看上去不夠奇妙，但它們遵循了高質量排版的已有規則，這些規則可以使得文檔易讀，同時看起來也非常舒適。

然而在一些情況下，LaTeX 也許並沒有提供適合你需要的命令或者環境，或者現有的命令產生的輸出和你想要的不同。

本章中，我將嘗試給出一些新的技巧，運用這些技巧可以教會 LaTeX 玩一些新的把戲，也可以使得 LaTeX 產生與眾不同的輸出。

6.1 新建命令、環境和宏包

你也許已經注意到我在這本書中介紹的所有命令都被包含在一個矩形框裡，並且出現在本書最後的索引中。我並沒有直接使用所需的 LaTeX 命令來實現這個，而是創建了一個宏包 (package)，並在其中定義了我所需要的命令和環境。現在我可以簡單的寫：

```
\begin{lscommand}
\ci{dum}
\end{lscommand}
```

\dum

在這個例子中，我使用了一個新的環境：lscommand。這個環境負責在命令的周圍畫出一個矩形框。同時我還使用了一個命令：\ci，這個命令負責輸出命令的名字，並且在索引中添加相應的條目。你可以在本書最後的索引中查找命令 \dum，然後你會發現有一個 \dum 的條目，這個條目中列出了包含有 \dum 命令的所有頁的頁碼。

一旦我不再喜歡在一個矩形框中排版命令，我可以輕鬆的改變 lscommand 環境的定義，來創建新的外觀。跟追蹤並修改所有使用原始的 LaTeX 命令在文字周圍畫框的地方相比，這種做法容易得多。

6.1.1 新建命令

爲了建立你自己的命令，可以使用如下的命令：

\newcommand{*name*}[*num*]{*definition*}

基本上，這個命令有兩個參量，第一個 *name* 是你想要建立的命令的名稱，

第二個 *definition* 是命令的定義。方括號裡的參數 *num* 是可選的，用於指定新命令所需的參量數目（最多 9 個）。如果不給出這個參數，默認就是 0，也就是新建的命令不要任何參量。

接下來的兩個例子有助你的理解。第一個例子定義了一個新的命令：`\tnss`。這個命令是句子 "The Not So Short Introduction to LaTeX 2ε" 的簡寫。如果你需要在文檔中多次使用本書的名稱，那麼定義這個命令將是非常方便的。

```
\newcommand{\tnss}{The not
    so Short Introduction to
    \LaTeXe}
This is ``\tnss'' \ldots{}
``\tnss''
```

This is "The not so Short Introduction to LaTeX 2ε" ... "The not so Short Introduction to LaTeX 2ε"

下一個例子演示了如何建立一個接受單一參數的命令。在命令的定義中，標記 `#1` 將被你指定的參量所代替。如果你想使用多個參量，那麼可以依次使用 `#2`、…… 、 `#9` 等標記。

```
\newcommand{\txsit}[1]
 {This is the \emph{#1} Short
      Introduction to \LaTeXe}
% in the document body:
\begin{itemize}
\item \txsit{not so}
\item \txsit{very}
\end{itemize}
```

- This is the *not so* Short Introduction to LaTeX 2ε
- This is the *very* Short Introduction to LaTeX 2ε

LaTeX 不允許你新建一個與現有命令重名的命令。如果你確實需要這麼做，有一個專門的命令用於處理這種情況：`\renewcommand`。它使用與命令 `\newcommand` 相同的語法。

在某些情況之下，你可能會希望使用 `\providecommand` 命令。它完成與 `\newcommand` 命令相同的工作。但如果命令已經存在，LaTeX 2ε 將會悄悄忽略原有的那個。

處理 LaTeX 命令後尾隨的空格有一些要注意的事項，參看第 4 頁可以獲得更多這方面的信息。

6.1.2 新建環境

與 `\newcommand` 命令類似，有一個命令用於建立新的環境。這個命令就是 `\newenvironment`，它的語法如下所示：

`\newenvironment{`*name*`}[`*num*`]{`*before*`}{`*after*`}`

同樣地，`\newenvironment` 命有一個可選的參量。在 *before* 中的內容將在此環境包含的文本之前處理，而在 *after* 中的內容將在遇到 `\end{`*name*`}` 命令時處理。

下面的例子演示了 `\newenvironment` 命令的用法：

```
\newenvironment{king}
{\rule{1ex}{1ex}%
     \hspace{\stretch{1}}}
{\hspace{\stretch{1}}%
     \rule{1ex}{1ex}}

\begin{king}
My humble subjects \ldots
\end{king}
```

■ My humble subjects . . . ■

參量 *num* 的使用方式與 \newcommand 命令相同。LaTeX 還同樣保證你不會不小心新建重名的環境。如果你確實希望改變一個現有的環境，你可以使用命令 \renewenvironment，它使用和命令 \newenvironment 相同的語法。

在這個例子中用到一些命令將在隨後解釋：\rule 命令的解釋可以參看第 96 頁， \stretch 命令的解釋可以參看第 91 頁，關於 \hspace 的信息可以在第 91 頁找到。

6.1.3 額外的空白間距

當創建新的環境時，你或許會爲遇到額外的空白間距而煩擾，這些間距可能產生嚴重的後果。比如當你建立一個標題環境，既不要自身的縮進也不要緊接著的下一段縮進時，在 begin 中加入命令 \ignorespaces 會使新環境忽略執行 begin 之後遇到的一切空白間距，而 end 就需要耍個小花招，因爲我們要等到環境結束後才開始處理。使用 \ignorespacesafterend，LaTeX 會在 end 處理完畢後，產生一個 \ignorespaces。

```
\newenvironment{simple}%
 {\noindent}%
 {\par\noindent}

\begin{simple}
See the space\\to the left.
\end{simple}
Same\\here.
```

See the space
to the left.

Same
here.

```
\newenvironment{correct}%
 {\noindent\ignorespaces}%
 {\par\noindent%
   \ignorespacesafterend}

\begin{correct}
No space\\to the left.
\end{correct}
Same\\here.
```

No space
to the left.

Same
here.

6.1.4 命令行的 LaTeX

如果使用類 Unix 的操作系統工作，你可能會在使用 Makefiles 建立你的 LaTeX 項目。那樣的話，用命令行參數操控 LaTeX 來創建同一份文檔的不同版本可是十分有趣的。如果把下列設定寫入你的文檔：

```
\usepackage{ifthen}
\ifthenelse{\equal{\blackandwhite}{true}}{
  % "black and white" mode; do something..
}{
  % "color" mode; do something different..
}
```

現在，你可以像這樣來操作 LaTeX：

```
latex '\newcommand{\blackandwhite}{true}\input{test.tex}'
```

首先，定義命令 `\blackandwhite`，然後使用 input 來讀入實際的文檔。要創建彩色版本文檔，需要設定 `\blackandwhite` 爲 false。

6.1.5 自建宏包

如果你定義了很多新的環境和命令，你的文檔的導言部分將變得相當長，在這種情況下，建立一個新的 LaTeX 宏包來存放所有你自己定義的命令和環境將是一個好的處理方式。然後你可以在文檔中使用 `\usepackage` 命令來引入自定義宏包。

```
% Demo Package by Tobias Oetiker
\ProvidesPackage{demopack}
\newcommand{\tnss}{The not so Short Introduction
                   to \LaTeXe}
\newcommand{\txsit}[1]{The \emph{#1} Short
                       Introduction to \LaTeXe}
\newenvironment{king}{\begin{quote}}{\end{quote}}
```

圖 6.1 – 宏包樣例。

寫一個宏包的基本工作就是將你原本很長的文檔導言內容拷貝到另外一個的文件中去，這個文件需要以 `.sty` 結尾。你還加入一個專用的命令：

`\ProvidesPackage{`*package name*`}`

這個命令應該放在你的宏包的最前面。`\ProvidesPackage` 告訴 LaTeX 宏包的名稱從而讓 LaTeX 在你嘗試兩次引入同一個宏包的時候給出一個明顯的錯誤信息，圖 6.1 給出了一個小的宏包示例，其中包含了我們之前定義的一些命令。

6.2 字體和字號

6.2.1 字體變換命令

LaTeX 根據文檔的邏輯結構（章節、脚註……）來選擇合適的字體和字體大小。在某些情況下，你可能會想要手動改變文檔使用的字體及其大小。爲了完成這個目的，你可以使用表 6.1 和表 6.2 中列出的那些命令。每個字體的實際大小是一個設計問題，並且它依賴於文檔所使用的文檔類及其選項。表 6.3 列出了這些命令在標準文檔類中的絕對 pt 大小。

```
{\small The small and
\textbf{bold} Romans ruled}
{\Large all of great big
\textit{Italy}.}
```

The small and **bold** Romans ruled all of great big *Italy*.

LaTeX 2_ε 的一個重要特徵是字體的各種屬性是相互獨立的，這意味著你可以改變字體的大小而仍然保留字體原有的粗體或者斜體的特性。

在**數學模式**中你可以使用字體變換命令來暫時退出**數學模式**，然後輸入一些正常的文字。如果你希望改變數學公式本身所使用的字體，LaTeX 提供了另外一套命令。參看表 6.4。

使用字體命令的時候，大括號 (curly braces) 扮演了一個重要角色。它們被用於建立所謂的**組** (group)。組限制了大多數 LaTeX 命令的作用範圍。

```
He likes {\LARGE large and
{\small small} letters}.
```

He likes large and small letters.

如果段落在字體的作用範圍中結束，那麼字號命令還將改變段落中行距。因此用於分組的反向大括號 } 不應該太早出現。注意下面兩個例子中 \par 命令的位置 [1]。

[1] \par 相當於一個空行

表 6.1 – 字體。

\textrm{...}	roman	\textsf{...}	sans serif
\texttt{...}	typewriter		
\textmd{...}	medium	\textbf{...}	**bold face**
\textup{...}	upright	\textit{...}	*italic*
\textsl{...}	*slanted*	\textsc{...}	Small Caps
\emph{...}	*emphasized*	\textnormal{...}	document font

表 6.2 – 字號。

\tiny	tiny font	\Large	larger font
\scriptsize	very small font	\LARGE	very large font
\footnotesize	quite small font	\huge	huge
\small	small font	\Huge	largest
\normalsize	normal font		
\large	large font		

表 6.3 – 標準文檔類中的絕對 pt 大小。

大小	10pt（默認）	11pt 選項	12pt 選項
\tiny	5pt	6pt	6pt
\scriptsize	7pt	8pt	8pt
\footnotesize	8pt	9pt	10pt
\small	9pt	10pt	11pt
\normalsize	10pt	11pt	12pt
\large	12pt	12pt	14pt
\Large	14pt	14pt	17pt
\LARGE	17pt	17pt	20pt
\huge	20pt	20pt	25pt
\Huge	25pt	25pt	25pt

表 6.4 – 數學字體。

\mathrm{...}	Roman Font
\mathbf{...}	**Boldface Font**
\mathsf{...}	Sans Serif Font
\mathtt{...}	Typewriter Font
\mathit{...}	*Italic Font*
\mathcal{...}	$\mathcal{CALLIGRAPHIC\ FONT}$
\mathnormal{...}	*Normal Font*

```
{\Large Don't read this!
 It is not true.
 You can believe me!\par}
```

{\Large Don't read this!
It is not true.
You can believe me!\par}

```
{\Large This is not true either.
But remember I am a liar.}\par
```

{\Large This is not true either.
But remember I am a liar.}\par

如果你希望改變整段甚至更多文本的字體，你可能應該使用字體變換命令的環境語法。

```
\begin{Large}
This is not true.
But then again, what is these
days \ldots
\end{Large}
```

\begin{Large}
This is not true.
But then again, what is these
days \ldots
\end{Large}

這將使你從一堆大括號中解脫出來。

6.2.2 戰戰兢兢，如履薄冰

正如本章開頭曾經說過的那樣，在你的文檔中直接運用這些命令來修改格式是非常危險的事情，因爲這種做法和 LaTeX 的基礎理念相反。在編寫 LaTeX 文檔的時候，要始終注意文章邏輯標記和樣式標識的分離。也就是如果你在文章的多個地方採用某種特殊的格式來排版一類經常使用的內容，就應該使用 \newcommand 來定義一個邏輯封裝命令，並通過這個命令來修改相應的表現格式。

```
\newcommand{\oops}[1]{%
 \textbf{#1}}
Do not \oops{enter} this room,
it's occupied by \oops{machines}
of unknown origin and purpose.
```

\newcommand{\oops}[1]{%
 \textbf{#1}}
Do not \oops{enter} this room,
it's occupied by \oops{machines}
of unknown origin and purpose.

這種方法具有一個明顯的優點，你可以在以後決定使用一些不是很有把握實現的特別外觀並使之不同於 \textbf，那時你就不需要遍歷你的整篇文章來找出所有 \textbf 的地方，然後一個一個地確定是不是要改成沒有把握的外觀。

6.2.3 建議

總結這一章中關於字體和字號的命令，下面是一個簡短的建議：

Remember! *The* MO RE fonts YOU use **in** a `document`, *the* more READABLE and *beautiful it becomes*.
記住！你使用的字體越多，文章看起來就越易讀越美觀。

6.3 間距

6.3.1 行距

如果你想在文檔中使用更大的行距，你可以在導言中使用如下命令進行設定：

`\linespread{`*factor*`}`

如 `\linespread{1.3}` 產生 1.5 倍行距，而 `\linespread{1.6}` 則產生雙倍行距。缺省情況下的行距爲 1。

注意 `\linespread` 的效果相當誇張而且不適合出版工作。因此如果你很想改變行距可能會希望使用如下的命令：

`\setlength{\baselineskip}{1.5\baselineskip}`

```
{\setlength{\baselineskip}%
           {1.5\baselineskip}
This paragraph is typeset with
the baseline skip set to 1.5 of
what it was before. Note the par
command at the end of the
paragraph.\par}

This paragraph has a clear
purpose, it shows that after the
curly brace has been closed,
everything is back to normal.
```

This paragraph is typeset with the baseline skip set to 1.5 of what it was before. Note the par command at the end of the paragraph.

This paragraph has a clear purpose, it shows that after the curly brace has been closed, everything is back to normal.

6.3.2 段落格式

在 LaTeX 中，有兩個參數可以影響段落的佈局。在文檔的導言部分，可以通過如下的定義來改變段落的佈局。

```
\setlength{\parindent}{0pt}
\setlength{\parskip}{1ex plus 0.5ex minus 0.2ex}
```

這兩個命令增加了段落間距，並將首行縮進設置爲 0。

例子中，長度設定中的 `plus` 和 `minus` 部分將使得 TeX 按照指定大小壓縮和伸展段落間距。爲了使得段落正確的顯示在頁面之上，TeX 將在 0.8ex 到 1.5ex 之間調整段落間距。

在歐洲大陸，段落通常用一些空白分隔並且一般首行不縮進。但是值得注意的是，這也會影響目錄。目錄的行距也會變得非常疏鬆。爲了避免這種情況，你可能需要將上面的兩個命令從導言中移到文檔中 `\tableofcontents` 以下適合的位置，或者根本不要使用這些，因爲一般來說專業的書籍都是用縮進並且通常不用空白來分離段落。如果你想縮進一個本來沒有縮進的段落[2]，可以在段落的開始使用命令：

`\indent`

當然，這個命令只有在 `\parindent` 不爲零的情況下才有效果。

[2]爲了縮進章節標題之後的第一個段落，可以使用 indentfirst 包。

爲了創建一個不縮進的段落，你可以在段落的開始部分使用命令：

`\noindent`

當文檔以正文而不是章節命令開始的時候，這個命令會提供方便。

6.3.3 水平間距

LaTeX 系統自動決定單詞和句子之間的距離。爲了增加水平距離，使用命令：

`\hspace{`*length*`}`

如果這個水平間距即使在行首或者行末也應該保持的話，用命令 `\hspace*` 代替 `\hspace`。命令的 *length* 參數在簡單的情況下只是一個帶有單位的數字。最重要的長度單位在表 6.5 中列了出來。

```
This\hspace{1.5cm}is a space
of 1.5 cm.
```

This is a space of 1.5 cm.

下面的命令

`\stretch{`*n*`}`

將產生一個特殊的橡皮長度：一個能把行內賸餘所有空隙填滿的空白。如果兩個 `\hspace{\stretch{`*n*`}}` 命令位於同一行，那麼它們將根據伸縮因子分配空間。

```
x\hspace{\stretch{1}}
x\hspace{\stretch{3}}x
```

x x x

當在正文中使用水平間距的時候，相對於字號來調整間距大小會更有道理。這可以通過使用與文本有關的單位 `em` 和 `ex` 來實現:

```
{\Large{}big\hspace{1em}y}\\
{\tiny{}tin\hspace{1em}y}
```

big y
tin y

6.3.4 垂直間距

在段落、節、小節…… 之間的距離是由 LaTeX 系統自動決定的。如果必要的話，可以在兩段之間增加額外的距離，使用的命令如下所示：

`\vspace{`*length*`}`

這個命令通常用於兩個空行之間。如果這個額外的行距應該在於頁的頂部和末尾也保留下來，那麼使用這個命令的星號版本 `\vspace*` 來代替 `\vspace`。

命令 `\stretch` 和 `\pagebreak` 結合使用可以在頁的最後一行輸出文本，也可以用來保證文本在頁面上垂直居中。

表 6.5 – TeX 單位。

mm	millimetre ≈ 1/25 inch	␣
cm	centimetre = 10 mm	␣␣
in	inch = 25.4 mm	␣␣␣
pt	point ≈ 1/72 inch ≈ $\frac{1}{3}$ mm	␣
em	approx width of an 'M' in the current font	␣
ex	approx height of an 'x' in the current font	␣

```
Some text \ldots

\vspace{\stretch{1}}
This goes onto the last line of the page.\pagebreak
```

同一段或同一個表格中兩行之間的距離可以用如下命令來指定：

`\\[`*length*`]`

使用命令 `\bigskip` 和 `\smallskip` 你可以獲得一個預定義的垂直間距。

6.4 頁面佈局

LaTeX 2_ε 允許你在 `\documentclass` 命令中指定紙張尺寸 (paper size)。然後它將自動的選擇合適的頁邊距。但有些時候你可能不滿意 LaTeX 的預設值，這個時候你可以自己改變這些參數。 圖 6.2 中顯示了所有能改變的頁面參數。這個圖是用 layout 宏包產生的[3]。

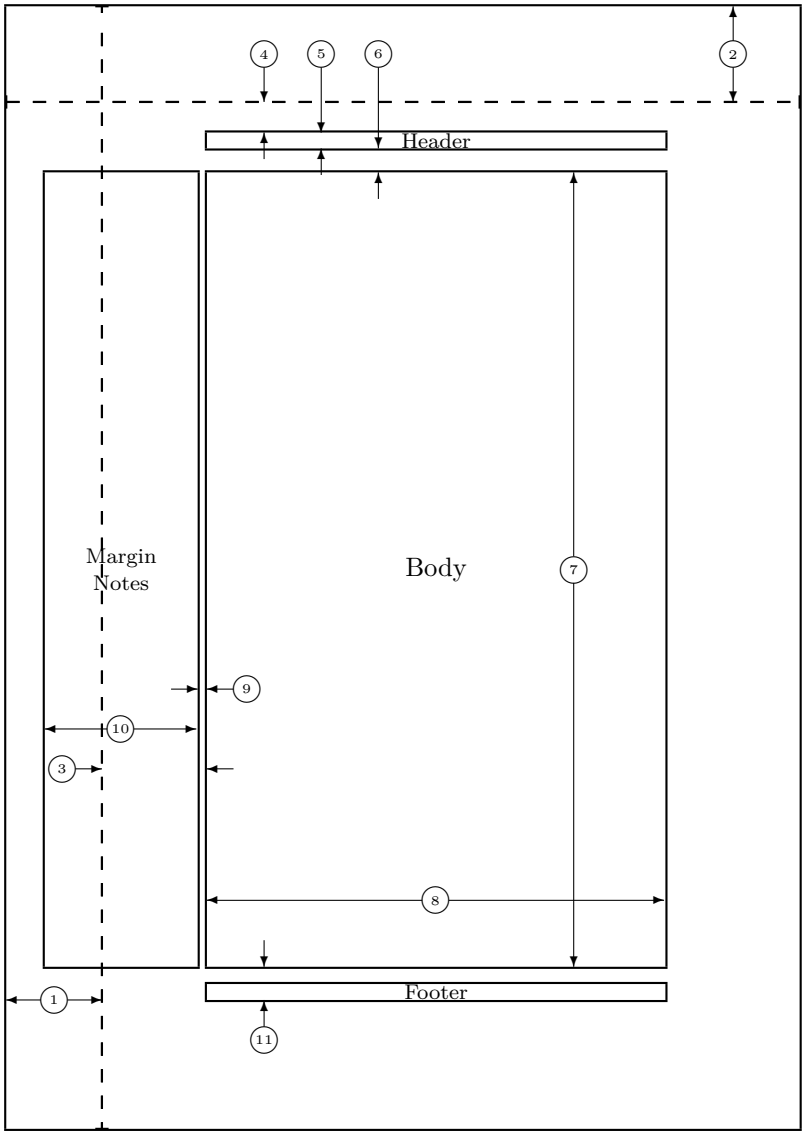
先等等! …… 在你開始幻想「讓這個狹窄的頁面看起來寬一點」之前，先花一些時間想想。和 LaTeX 中的大多數規定一樣，缺省的頁面佈局是有其合理原因的。

確實，相對於你的 MS Word 頁面來說，它看上去非常的狹窄。但是看看你喜歡的書籍 [4]並且統計每個標準文本行的字符數目。你會發現每行的字符不超過 66 個。現在你的 LaTeX 頁面也正是如此。經驗顯示，如果在一行中塞入更多的字符，閱讀將變得困難。這是因爲眼睛從行的開始移動到行的結束變得困難了。這也是報紙爲何要排版成多欄形式的原因。

因此如果你決定增加版芯的寬度，頭腦中要明白你正在使給你的讀者製造困難。警告已經說的夠多了，接下來我將告訴你如何去做……

LaTeX 提供了兩個命令來改變這些參數。他們通常在文章的導言部分使用。

[3] `macros/latex/required/tools`
[4] 我說的是卓有聲譽的出版商正式出版的書籍。



1	one inch + \hoffset	2	one inch + \voffset
3	\oddsidemargin = 28pt or \evensidemargin	4	\topmargin = 23pt
5	\headheight = 12pt	6	\headsep = 18pt
7	\textheight = 598pt	8	\textwidth = 345pt
9	\marginparsep = 7pt	10	\marginparwidth = 115pt \marginparpush = 5pt (not shown)
11	\footskip = 25pt \hoffset = 0pt \paperwidth = 597pt		\voffset = 0pt \paperheight = 845pt

圖 6.2 – 頁面佈局參數。

第一個命令給某些參數一個固定的值：

\setlength{*parameter*}{*length*}

第二個命令給某些參數增加一個長度：

\addtolength{*parameter*}{*length*}

第二個命令實際上比 \setlength 命令更爲實用，因爲你可以相對於現有的設置來獲得所需的結果。爲了給文本的寬度增加 1 釐米，我將如下的命令放置到文檔導言：

```
\addtolength{\hoffset}{-0.5cm}
\addtolength{\textwidth}{1cm}
```

這時候，你可能會想要看看 calc 包，它允許你在 \setlength 的參量中進行算術運算。它也可以運用到任何用數值作爲函數參量的地方。

6.5 更有趣的長度

只要可能，就應該避免在 LaTeX 文檔中使用絕對長度。我更願意通過頁面中其他元素的寬度或高度來指定長度。比如一個圖形，我指定 \textwidth 作爲它的寬度從而使得圖形恰好充滿整個頁面。

下面的三個命令允許你獲得一個文本串的寬度、高度以及深度。

\settoheight{*variable*}{*text*}
\settodepth{*variable*}{*text*}
\settowidth{*variable*}{*text*}

下面的例子顯示了如何應用這些命令：

```
\flushleft
\newenvironment{vardesc}[1]{%
  \settowidth{\parindent}{#1:\ }
  \makebox[0pt][r]{#1:\ }}{}

\begin{displaymath}
a^2+b^2=c^2
\end{displaymath}

\begin{vardesc}{Where}$a$,
$b$ -- are adjoin to the right
angle of a right-angled triangle.

$c$ -- is the hypotenuse of
the triangle and feels lonely.

$d$ -- finally does not show up
here at all. Isn't that puzzling?
\end{vardesc}
```

$$a^2 + b^2 = c^2$$

Where: a, b – are adjoin to the right angle of a right-angled triangle.

c – is the hypotenuse of the triangle and feels lonely.

d – finally does not show up here at all. Isn't that puzzling?

6.6 盒子

LaTeX 使用盒子來建立頁面。首先，每個字符都是一個小的盒子，這些盒子粘結起來構成單詞，單詞粘結起來構成一行。值得注意的是，單詞之間粘結的是一種特殊的「膠水」(glue)，它是有彈性的，可以使 LaTeX 壓縮或者延伸使得單詞將恰好構成頁面的一行。

我承認，這裡的描述是實際情況一個極度簡化了的版本，但關鍵在於 TeX 對盒子和膠水進行操作。不是只有字母才能成爲盒子，你幾乎可以把任何東西包括其他盒子放到一個盒子中。然後 LaTeX 將會像處理單個字母一樣處理這個盒子。

在過去的章節中，儘管我並沒有明確的說出來，你已經遇到了一些盒子。例如 `tabular` 環境和 `\includegraphics` 命令就都產生了一個盒子。這就意味著你可以輕鬆的將兩個表格或圖像並列。你唯一需要保證的就是它們寬度的總和不大於文本寬度。

使用如下命令可以把一個段落放置到盒子中：

`\parbox[`*pos*`]{`*width*`}{`*text*`}`

或者用下面這個環境完成同樣的事情：

`\begin{minipage}[`*pos*`]{`*width*`}` text `\end{minipage}`

參數 `pos` 可以取以下字符中的一個 `c`、`t` 或 `b`，這個參數用於控制盒子相對周圍文本基線的垂直方向對齊。`width` 是一個長度參量用於調整盒子的寬度。`minipage` 和 `\parbox` 的區別在於你可能無法在一個 `parbox` 中使用所有的命令或者環境，而幾乎任何東西都可以在 `minipage` 中使用。

雖然`\parbox` 可以打包整個段落，完成分行在內的幾乎所有事情， LaTeX 中還存在與此不同的另外一類盒子命令用於處理水平對齊的東西。我們已經知道其中的一個 —— `\mbox`，它只是簡單地將其它盒子包含在一個盒子裡，從而防止 LaTeX 斷開兩個單詞。因爲盒子中可以包含盒子，它可以給予作者幾乎無限的靈活性。

`\makebox[`*width*`][`*pos*`]{`*text*`}`

`width` 定義了生成的盒子的外部寬度[5]。除了長度表達式，你也可以傳遞 `\width`、 `\height`、 `\depth` 和 `\totalheight` 給 `width`。這幾個值是測量盒子內部文本來獲得的。參數 *pos* 接受一個字符值： **c** – 居中、**l** – 靠左、**r** – 靠右和 **s** – 將文本均勻分佈到整個盒子中。

命令 `\framebox` 和 `\makebox` 完成同樣的工作，不同之處在於它在內部文本的周圍畫出一個矩形框。

下面的例子演示了你使用命令 `\makebox` 和 `\framebox` 能完成的一些工作：

[5]這意味著在盒子內部看來，盒子的寬度可能會小一些，你甚至可以將盒子的寬度設置爲 0 pt，這樣可以使得盒子中的內容不影響其外部的佈局。

```
\makebox[\textwidth]{%
    c e n t r a l}\par
\makebox[\textwidth][s]{%
    s p r e a d}\par
\framebox[1.1\width]{Guess I'm
    framed now!} \par
\framebox[0.8\width][r]{Bummer,
    I am too wide} \par
\framebox[1cm][l]{never
    mind, so am I}
Can you read this?
```

c e n t r a l

s p r e a d

Guess I'm framed now!

Bummer, I am too wide

never mind, so am I Can you read this?

現在我們已經知道怎麼控制盒子的水平方向長度了，接下來的步驟是學習如何控制垂直方向[6]。對於 LaTeX 來說，輕而易舉。命令

\raisebox{*lift*}[*extend-above-baseline*][*extend-below-baseline*]{*text*}

讓你能夠定義一個盒子在垂直方向的屬性。在前面的三個參數中，你可以使用 \width、\height、\depth 和 \totalheight，這樣可以使得盒子的參數能夠與盒子內部的文本匹配。

```
\raisebox{0pt}[0pt][0pt]{\Large%
\textbf{Aaaa\raisebox{-0.3ex}{a}%
\raisebox{-0.7ex}{aa}%
\raisebox{-1.2ex}{r}%
\raisebox{-2.2ex}{g}%
\raisebox{-4.5ex}{h}}}
he shouted but not even the next
one in line noticed that something
terrible had happened to him.
```

Aaaaaaargh he shouted but not even the next one in line noticed that something terrible had happened to him.

6.7 標尺和支撐

很多頁之前你可能注意到這樣的命令：

\rule[*lift*]{*width*}{*height*}

通常它被用來輸出一個簡單的黑色盒子。

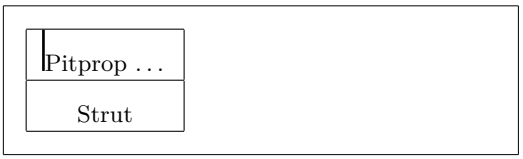
```
\rule{3mm}{.1pt}%
\rule[-1mm]{5mm}{1cm}%
\rule{3mm}{.1pt}%
\rule[1mm]{1cm}{5mm}%
\rule{3mm}{.1pt}
```

這個命令可以用來產生水平方向和垂直方向的線條。例如標題頁上的直線就是用一個 \rule 命令創建的。

一種特殊的例子是沒有寬度只有高度的標尺。在專業的出版中，這被稱爲支撐 (Struts)。它被用來保證頁面的某個元素具有一個確定的高度最小值。你可以在 tabular 環境中使用支撐來使得某行具有一個特定的高度最小值。

[6] 全面控制僅僅是水平方向控制和垂直方向控制的同時運用……

```
\begin{tabular}{|c|}
\hline
\rule{1pt}{4ex}Pitprop \ldots\\
\hline
\rule{0pt}{4ex}Strut\\
\hline
\end{tabular}
```



全篇結束。

參考文獻

[1] Leslie Lamport. *LaTeX: A Document Preparation System.* Addison-Wesley, Reading, Massachusetts, second edition, 1994, ISBN 0-201-52983-1.

[2] Donald E. Knuth. *The TeXbook,* Volume A of *Computers and Typesetting*, Addison-Wesley, Reading, Massachusetts, second edition, 1984, ISBN 0-201-13448-9.

[3] Frank Mittelbach, Michel Goossens, Johannes Braams, David Carlisle, Chris Rowley. *The LaTeX Companion, (2nd Edition).* Addison-Wesley, Reading, Massachusetts, 2004, ISBN 0-201-36299-6.

[4] Michel Goossens, Sebastian Rahtz and Frank Mittelbach. *The LaTeX Graphics Companion.* Addison-Wesley, Reading, Massachusetts, 1997, ISBN 0-201-85469-4.

[5] Each LaTeX installation should provide a so-called *LaTeX Local Guide*, which explains the things that are special to the local system. It should be contained in a file called `local.tex`. Unfortunately, some lazy sysops do not provide such a document. In this case, go and ask your local LaTeX guru for help.

[6] LaTeX3 Project Team. *LaTeX 2ε for authors.* Comes with the LaTeX 2ε distribution as `usrguide.tex`.

[7] LaTeX3 Project Team. *LaTeX 2ε for Class and Package writers.* Comes with the LaTeX 2ε distribution as `clsguide.tex`.

[8] LaTeX3 Project Team. *LaTeX 2ε Font selection.* Comes with the LaTeX 2ε distribution as `fntguide.tex`.

[9] D. P. Carlisle. *Packages in the 'graphics' bundle.* Comes with the 'graphics' bundle as `grfguide.tex`, available from the same source your LaTeX distribution came from.

[10] Rainer Schöpf, Bernd Raichle, Chris Rowley. *A New Implementation of LaTeX's verbatim Environments.* Comes with the 'tools' bundle as `verbatim.dtx`, available from the same source your LaTeX distribution came from.

[11] Vladimir Volovich, Werner Lemberg and LaTeX3 Project Team. *Cyrillic languages support in LaTeX.* Comes with the LaTeX 2ε distribution as `cyrguide.tex`.

[12] Graham Williams. *The TeX Catalogue* is a very complete listing of many TEX and LATEX related packages. Available online from CTAN:/tex-archive/help/Catalogue/catalogue.html

[13] Keith Reckdahl. *Using EPS Graphics in LATEX 2ε Documents*, which explains everything and much more than you ever wanted to know about EPS files and their use in LATEX documents. Available online from CTAN:/tex-archive/info/epslatex.ps

[14] Kristoffer H. Rose. *XY-pic User's Guide*. Downloadable from CTAN with XY-pic distribution

[15] John D. Hobby. *A User's Manual for MetaPost*. Downloadable from http://cm.bell-labs.com/who/hobby/

[16] Alan Hoenig. *TEX Unbound*. Oxford University Press, 1998, ISBN 0-19-509685-1; 0-19-509686-X (pbk.)

[17] Urs Oswald. *Graphics in LATEX 2ε*, containing some Java source files for generating arbitrary circles and ellipses within the `picture` environment, and *MetaPost - A Tutorial*. Both downloadable from http://www.ursoswald.ch