LaTeX Notes v 1.17

Alpha Huang[1]

2008年6月2日

[1] http://www.dralpha.com/

看什麼看，沒見過空白頁？

序

滿紙荒唐言，一把辛酸淚！都雲作者痴，誰解其中味？[1]

— 曹雪芹

最早聽說 LaTeX 大約是 2002 年，一位同事演示了用它排版的一篇文章和幾幅圖。包老師[2]不以爲然，因爲那些東西用 Microsoft Word 和 Visio 也可以做到，而且可以做得更快。再次聽說它是王垠同學在鬧退學，傳說他玩 Linux 和 LaTeX 而走火入魔。

大約是 2005 年底，看了一下 lshort，用 LaTeX 記了些數學筆記，開始有點感覺。包老師生性愚鈍，所以喜歡相對簡單的東西。HTML、Java 都用手寫，FrontPage、Dreamweaver、JBuilder 之類笨重的傢伙看兩眼就扔了，所以喜歡上 LaTeX 只是時間問題。

次年老妻要寫博士論文，拿出 Word 底稿讓我排版。大家都知道 Word 太簡單了，誰都能用，但是不是誰都能用好。人稱電腦殺手的老妻製作的 Word 文檔自然使出了各種奇門遁甲，加上她實驗室、學校和家裡電腦裡的三個 EndNote 版本互不兼容，實在難以馴服。我只好重起爐灶，拿她的博士論文當小白鼠，試驗一下 LaTeX 的威力。

就這樣接觸了兩三年，總算略窺門徑，感覺 LaTeX 實在是博大精深，浩如煙海。而人到中年大腦儲存空間和處理能力都有點捉襟見肘，故時常作些筆記。一來對常用資料和問題進行彙編索引，便於查詢；二來也記錄一些心得。

---

[1] 當年作博士論文時雖不曾增刪五次披閱十載，也被折磨得欲仙欲死，故與室友戲言將此五絕加入序言。多年以後的今天終於實現了此夙願。

[2] 吾有多重人格，比如本色的是阿黃，下圍棋的是隱忍灰衣人，道貌岸然的是包老師。

日前老妻吵著要學 LaTeX，便想這份筆記對初學者或有些許借鑑意義，於是系統地整理了一番，添油加醋包裝上市。

原本打算分九章，以紀念《九章算術》，實際上第八章完成時已如強弩之末，最後一章還須另擇黃道吉日。

本文第一章談談歷史背景；第二章介紹入門基礎；第三至五章講解數學、插圖、表格等對象的用法；第六章是一些特殊功能；第七、八章討論中文和字體的處理；第九章附加定製內容。

從難易程度上看前兩章較簡單，插圖、字體兩章較難。一般認爲 LaTeX 相對於微軟的傻瓜型軟件比較難學，所以這裡採取循序漸進，溫水煮青蛙的方法。

初則示弱，麻痺讀者；再則巧言令色，請君入甕；三則舌綻蓮花，誘敵深入；彼入得罄中則摧動機關，關門打狗；繼而嚴刑拷打，痛加折磨；待其意亂情迷徬徨無計之時，給予當頭棒喝醍醐灌頂，雖戛然而止亦餘音繞樑。

鄙人才疏學淺功力不逮，面對汗牛充棟罄竹難書[3]的資料，未免考慮不周掛一漏萬，或有誤導，敬請海涵。若有高手高手高高手略撥閒暇指點一二，在下感激不盡[4] 。

借此感謝一下老妻，如果不是伊天天看韓劇，包老師也不會有時間灌水和整理這份筆記。

[3]此處用法循阿扁古例。

[4]huang.xingang@gmail.com

目 錄

看什麼看，沒見過空白頁？

第一章　簡介

滾滾長江東逝水，浪花淘盡英雄。是非成敗轉頭空。青山依舊在，幾度夕陽紅。

白髮漁樵江渚上，慣看秋月春風。一壺濁酒喜相逢。古今多少事，都付笑談中。

— 楊慎《臨江仙》

1.1　歷史回顧

LaTeX 是一種面向數學和其它科技文檔的電子排版系統。一般人們提到的 LaTeX 是一個總稱，它包括 TeX、LaTeX、$\mathcal{AMS}$-LaTeX 等[1]。

TeX 的開發始於 1977 年 5 月，Donald E. Knuth[2]開發它的初衷是用於《The Art of Computer Programming》的排版。1962 年 Knuth 開始寫一本關於編譯器設計的書，原計劃是 12 章的單行本。不久 Knuth 覺得此書涉及的領域應該擴大，於是越寫越多，如滔滔江水連綿不絕，又如黃河氾濫一發不可收拾。1965 年完成的初稿居然有 3000 頁，全是手寫的！據出版商估計，這些手稿印刷出來需要 2000 頁，出書的計劃只好改爲七卷，每卷一或兩章。1976 年 Knuth 改寫第二卷的第二版時，很鬱悶地發現第一卷的鉛版不見了，而當時電子排版剛剛興起，質量還差強人意。於是 Knuth 仰天長嘯：「我要扼住命運的咽喉」，決定自己開發一個全新的系統，這就是 TeX。

[1] 一般認爲 TeX 是一種引擎，LaTeX 是一種格式，而 $\mathcal{AMS}$-LaTeX 等是宏集。此處目的是簡介，故不展開討論。

[2] 斯坦福大學計算機系教授，已退休。

1978 年 TEX 第一版發佈後好評如潮，Knuth 趁熱打鐵在 1982 年發佈了第二版。人們現在使用的 TEX 基本就是第二版，中間只有一些小的改進。1990 年 TEX v3.0 發佈後，Knuth 宣佈除了修正 bug 外停止 TEX 的開發，因爲他要集中精力完成那本巨著的後幾卷[3]。此後每發佈一個修正版，版本號就增加一位小數，使得它趨近於π（目前是 3.141592）。Knuth 希望將來他離世時，TEX 的版本號永遠固定下來，從此人們不再改動他的代碼。他開發的另一個軟件 METAFONT 也作類似處理，它的版本號趨近於e，目前是 2.71828。

TEX 是一種語言也是一個宏處理器，這使得它很好很強大，但是它同時又很繁瑣，讓人難以接近。因此 Knuth 提供了一個對 TEX 進行了封裝的宏集 Plain TEX，裡面有一些高級命令，有了它最終用戶就無須直接面對枯燥無味的 TEX。

然而 Plain TEX 還是不夠高級，所以 Leslie Lamport[4]在 80 年代初期開發了另一個基於 TEX 的宏集 LATEX。1992 年 LATEX v2.09 發佈後，Lamport 退居二線，之後的開發活動由 Frank Mittelbach 領導的 The LaTeX Team 接管。此小組發佈的最後版本是 1994 年的 LATEX 2_ε，他們同時還在進行 LATEX 3 的開發，只是正式版看起來遙遙無期。

起初，美國數學學會（American Mathematical Society，AMS）看著TEX 是好的，就派 Michael Spivak 寫了 $\mathcal{AMS}$-TEX，這項基於 Plain TEX 的開發活動進行了兩年（1983–1985）。後來與時俱進的 AMS 又看著 LATEX 是好的，就想轉移陣地，但是他們的字體遇到了麻煩。恰好 Mittelbach 和 Rainer Schöpf（後者也是 LaTeX Team 的成員）剛剛發佈了 New Font Selection Scheme for LATEX（NFSS），AMS 看著還不錯，就拜託他們把 AMSFonts加入 LATEX，繼而在 1989 年請他們開發 $\mathcal{AMS}$-LATEX。$\mathcal{AMS}$-LATEX 發佈於 1990 年，之後它被整合爲 $\mathcal{AMS}$ 宏包，像其它宏包一樣可以直接運行於 LATEX。

[3]已出版的前三卷是：《Fundamental Algorithms》、《Seminumerical Algorithms》、《Sorting and Searching》；第四卷《Combinatorial Algorithms》和第五卷《Syntactic Algorithms》正在寫作中，預計 2015 年出版；第六卷《Theory of Context-free Languages》和第七卷《Compiler Techniques》尚未安排上工作日程。

[4]現供職於微軟研究院。

1.2 優點和缺點

當前的文字處理系統大致可以分爲兩種：標記語言（Markup Language）式的，比如LaTeX；所見即所得（WYSIWYG）式的，比如 MS Word[5]。

一般而言，LaTeX 相對於所見即所得系統有如下優點：

- 高質量 它製作的版面看起來更專業，數學公式尤其賞心悅目。
- 結構化 它的文檔結構清晰。
- 批處理 它的源文件是文本文件，便於批處理，雖然解釋（parse）源文件可能很費勁。
- 跨平台 它幾乎可以運行於所有電腦硬件和作業系統平台。
- 免費 多數 LaTeX 軟件都是免費的，雖然也有一些商業軟件。

相應地，LaTeX 的工作流程、設計原則，資源的缺乏，以及開發人員的歷史局限性等種種原因也導致了一些缺陷：

- 製作過程繁瑣，有時需要反覆編譯，不能直接或實時看到結果。
- 宏包魚龍混雜，水準參差不齊，風格不夠統一。
- 排版風格比較統一，但因而缺乏靈活性。
- 用戶支持不夠好，文檔不完善。
- 對國際語言和字體的支持很差。

拋開 MS Word 不談，即使跟同爲標記語言的 HTML/Web 系統相比，LaTeX 也有一些不足之處。比如 Web 瀏覽器對 HTML 內容的渲染（render）比 DVI 瀏覽器對 LaTeX 內容的渲染要快上許多，基本上可以算是實時。雖然 HTML 內容可能沒有 LaTeX 那麼複雜，但是 DVI 畢竟是已經被 LaTeX 編譯過的格式。

[5]其實 Word 也有自己的標記語言域代碼（field code），只是一般用戶不瞭解。

還有一點令人困惑的是，有一部分 LaTeX 陣營的人士習慣於稱對方爲「邪惡的」或「出賣靈魂的」，如果昂貴的微軟系統應當爲人詬病，那麼更貴的蘋果係統爲何卻被人追捧？

2000 年有記者在採訪 Lamport 時問：「爲什麼當前沒有高質量的所見即所得排版系統？」他回答道：「門檻太高了，一個所見即所得系統要做到 LaTeX 當前的水平，工作量之大不是單槍匹馬所能完成[6]。微軟那樣的大公司可以做，但是市場太小了。我偶爾也會想加入「Dark Side」，讓微軟給我一組人馬來開發一個這樣的系統。」（包老師註：他果然於次年加入微軟。）

竊以爲這兩大陣營其實是蘿蔔青菜的關係，與其抱殘守缺、互相攻訐，不如各取所需；甚至可以捐棄前嫌、取長補短，共建和諧社會。

1.3 軟件準備

LaTeX 是一個軟件系統，同時也是一套標準。遵照這些標準，實現了（implement）所要求功能的軟件集合被稱爲發行版（distribution）。與此類似的例子有 Java 和 Linux，比如SUN、IBM、BEA 等公司都有自己的 Java 虛擬機（JVM），它們都被稱作 Java 的實現；而Linux 有 Red Hat/ Fedora、Ubuntu、SuSE 等衆多的發行版。

表 1.1: LaTeX 發行版與編輯器

作業系統	發行版	編輯器
Windows	MikTeX	TeXnicCenter、WinEdt
Unix/Linux	TeX Live	Emacs、vim、Kile
Mac OS	MacTeX	TeXShop

LaTeX 發行版只提供了一個 LaTeX 後台處理機制，用戶還需要一個前台編輯器來編輯它的源文件。常用的 LaTeX 發行版和編輯器見表 1.1。在使用 LaTeX 的過程中可能還需要其它一些軟件，將在後面相關章節中分別介紹。

[6] TeX/LaTeX 也不單單是那幾個大腕兒完成的，他們背後還有衆多默默無聞的小人物，比如當年 Knuth 手下的大批學生。此所謂一將功成萬骨枯。

1.4 學習方法

> 在科學上沒有平坦的大道，只有那些不畏勞苦沿著陡峭山路攀登的人，才有希望達到光輝的頂點。
>
> — 卡爾 馬克思

> 無他，唯手熟爾。
>
> — 賣油翁

> 用心。
>
> — 斯蒂芬 周

限於篇幅和水平，本文只能提供一個概覽外加一些八卦。比較嚴謹的入門資料有 Tobias Oetiker 的《A (Not So) Short Introduction to LaTeX 2_ε》[1]（簡稱lshort）；若想對 LaTeX 有更深入全面的瞭解，可以拜讀 Mittelbach 的《The LaTeX Companion》[2]。

中文資料可參考李果正的《大家來學 LaTeX》[3]，lshort 有吳凌雲等人翻譯的中文版本[7]。

Comprehensive TeX Archive Network（CTAN）和 TeX Users Group（TUG）提供了權威、豐富的資源。

英國TUG 和 CTeX 分別提供了常見問題集（FAQ）[4;5]，一般問題多會在這裡找到答案。

中文 TeX 論壇有水木清華 BBS TeX 版、CTeX 論壇。

參考文獻

[1] Tobias Oetiker. *A (Not So) Short Introduction to LaTeX2e*, 2008. URL http://www.ctan.org/tex-archive/info/lshort/english/.

[2] Frank Mittelbach, Michel Goossens, Johannes Braams, David Carlisle, and Chris Rowley. *The LaTeX Companion (Tools and Techniques for*

[7] 此譯本首發於 CTeX 論壇，但是需要註册才能看見鏈接，所以請讀者自行搜索。

Computer Typesetting). Addison-Wesley, 2nd edition, 2004. URL http://www.amazon.com/exec/obidos/tg/detail/-/0201362996/.

[3] 李果正. 大家來學*LaTeX*, 2004. URL http://edt1023.sayya.org/tex/latex123/.

[4] UK TeX User Group. UK List of TeX Frequently Asked Questions. URL http://www.tex.ac.uk/faq/.

[5] 中國TeX用戶組. CTeX常見問題集, 2005. URL http://www.ctex.org/CTeXFAQ/.

第二章　入門

2.1　Hello, World!

把下面例子用編輯器保存爲 `hello_world.tex`，這就是一個最簡單的 LaTeX 源文件。

```
%hello_world.tex
\documentclass{article}
\begin{document}
    Hello, World!
\end{document}
```

有了源文件，我們可以在命令行把它編譯成 DVI 文件（DVI 格式見2.2.1小節）。此命令知道輸入的是 LaTeX 源文件，所以這裡的 `.tex` 後綴可以省略。以後的示例中可以省略的後綴都用 () 標出，不再特別聲明。

```
latex hello_world(.tex)
```

如果係統顯示類似下面的錯誤信息，請檢查源文件是否有拼寫錯誤。`.log` 文件裡有更詳細的編譯信息。

```
! LaTeX Error:
...
! Emergency stop.
...
No pages of output.
Transcript written on hello_world.log.
```

如果編譯成功，系統會報出類似下面的信息：

```
Output written on hello_world.dvi (1 page, 232 bytes).
Transcript written on hello_world.log.
```

每種 LaTeX 發行包附帶不同的 DVI 瀏覽器，比如 MiKTeX 的是 yap。

```
yap hello_world(.dvi)
```

2.2 格式及其轉換

2.2.1 頁面描述語言

頁面描述語言（Page Description Language，PDL）是一種在較高層次上描述實際輸出結果的語言。本文只討論其中三種與 LaTeX 緊密相關的格式：DVI、PostScript、PDF。

PostScript

最早的打印機只用於打印字符，它使用的硬字符與打字機類似。後來出現的點陣（dot matrix）打字機用一系列的點來「畫」出字符，當然它也可以畫出圖形。當時向量圖的打印只能由繪圖儀（plotter）來完成。

1976 年，施樂（Xerox）推出了首台激光打印機，它結合了點陣打印機和繪圖儀的優點，可以同時打印高質量的圖形和文字。

同一時期，John Warnock 也在醞釀一種類似於 Forth 的圖形設計語言，也就是後來的 PostScript（PS），當時他正在舊金山一家電腦圖形公司 Evans & Sutherland 工作。1978 年老闆想讓 Warnock 搬到位於猶他州的總部，他不想搬家就跳槽到了施樂。

Warnock 和 Martin Newell 開發了新的圖形系統 JaM（John and Martin），它後來被合併到施樂的打印機驅動程序 InterPress 中去。這兩位還開發過另一個系統 MaJ。

1982 年，Warnock 和施樂研究中心圖形實驗室主任 Chuck Geschke 一起離開施樂，成立了 Adobe 公司。Newell 後來也加入了 Adobe。

1984 年 Adobe 發佈 PS 後不久，Steve Jobs 跑來參觀，並建議用它來驅動激光打印機。次年，武裝著 PS 驅動的 Apple LaserWriter 橫空出世，打響了 80 年代中期桌面出版革命的第一槍。

90 年代中後期，廉價噴墨打印機的流行使得 PS 逐漸式微，因爲 PS 驅動對它們畢竟是一個成本負擔。

PDF

1993 年，Adobe 推出了一種開放的格式：Portable Document Format（PDF），它於 2007 年成爲 ISO 32000 標準。除了開放，PDF 比起 PS 還有一些其它優勢：

- PDF 基本上是 PS 的一個子集，因此更輕便。
- PDF 可以嵌入更先進的字體，具體見8.2節。
- PDF 支持嵌入亂七八糟的東東，比如動畫。
- PDF 支持透明圖形。

PDF 雖然擁有上述優勢，起初它的推廣卻並不順利，因爲其讀寫工具 Acrobat 太貴。Adobe 很快推出了免費的 Acrobat Reader（後更名爲 Adobe Reader），並不斷改進 PDF，終於使它超越了曾經的事實標準 PS，成爲網絡時代電子文檔的新標準。

DVI

Knuth 最初設計的 TEX 只能用於 XGP 打印機，這台打印機本身還需要一台 PDP-6 主機爲它服務。1979 年，David Fuchs[1]提出把 TEX 的輸出改爲設備無關的格式，也就是 Device Independent format（DVI）。

DVI 只是一種中間格式，用戶還需要另外的處理程序（driver）把它轉換爲其它格式，比如 PS 或 PDF ，甚至 PNG、SVG 等。DVI 不能嵌入字體和圖形，PS 和 PDF 可以選擇是否嵌入字體。

[1]Fuchs 本科畢業於普林斯頓，1978 年進入斯坦福攻讀博士學位。他不是 Knuth 的學生，但是完成過一些 TEX 的開發任務。他在 Adobe 工作過一段時間，現在混入了娛樂圈，擔任過電影《Red Diaper Baby》和《Haiku Tunnel》的製片人。

Ghostscript

PS 輸出時需要一個解釋器（Raster Image Processor，RIP）來把它轉換爲點陣圖形。RIP 可以是軟件，也可以是固件（firmware）或硬件[2]。

Ghostscript 是一個基於 RIP 的軟件包，除了 RIP 它還有一些其它功能，比如處理 EPS，把 PS 轉換爲 PDF 等。Ghostscript 已經被移植到 Windows、Unix/Linux、Mac OS 等多種作業系統，和它匹配的前端圖形用戶界面（GUI）有GSview、Ghostview、gv等。

2.2.2 格式轉換

DVI、PS、PDF 等格式的的轉換關係如圖 2.1所示。

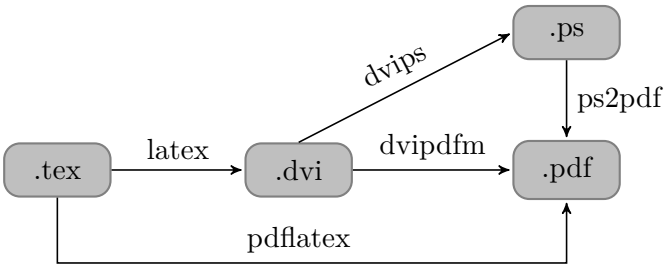


圖 2.1: 格式轉換

最早的 driver 是 `dvips`，它把 DVI 轉換爲 PS。`dvipdf` 把 DVI 轉爲 PDF，它後來被 `dvipdfm` 所取代；`dvipdfm` 主要用於處理單字節字符，1999 年之後停止開發；在 `dvipdfm` 基礎上發展來的 `dvipdfmx` 可以處理多字節編碼（字符編碼詳見7.1節）。

pdfTEX 是一種特殊的driver，它跳過 DVI，直接用 TEX 源文件生成 PDF。基於 pdfTEX 的 pdfLATEX 則把LATEX 源文件轉爲 PDF。

包老師傾向於 `dvipdfmx`，因爲它對圖形格式的兼容性較好，而且擅長處理中文。

得到 DVI 後，我們可以在控制台用以下命令把它轉爲 PDF。

```
dvipdfm hello_world(.dvi)
```

[2]固件 RIP 在打印機內置處理器上運行，硬件 RIP 常見於高端打印設備。

我們也可以把它轉爲 PS，接著用 Ghostscript 的一個命令行程序把它轉換爲 PDF，注意第二個命令需要 .ps 後綴。一般情況下不推薦這種方法，因爲它多了個步驟。

```
dvips hello_world(.dvi)
ps2pdf hello_world.ps
```

pdfLaTeX 用法如下。

```
pdflatex hello_world(.tex)
```

2.3 LaTeX 語句

LaTeX 源文件的每一行稱作一條語句（statement），語句可以分三種：命令（command）、數據（data）和註釋（comment）。

命令分爲兩種：普通命令和環境（environment）。普通命令以\ 起始，大多隻有一行；而環境包含一對起始聲明和結尾聲明，用於多行的場合。命令和環境可以互相嵌套。

數據就是普通內容。註釋語句以 % 起始，它在編譯過程中被忽略。

例如在2.1節例1中，第一行是註釋，第二行是普通命令；第三、五行是環境的起始和結尾聲明；第四行是數據。

2.4 文檔結構

2.4.1 文檔類、序言、正文

LaTeX 源文件的結構分三大部分，依次爲：文檔類聲明、序言（可選）、正文。

文檔類聲明用來指定文檔的類型；序言（preamble）用來完成一些特殊任務，比如引入宏包，定義命令，設置環境等；文檔的實際內容則放在正文部分。這裡的正文指得是\begin{document}和\end {document}之間的部分，和通常人們心目中的「正文」概念有所出入。

這三部分的基本語法如下：

```
\documentclass[options]{class}  %文檔類聲明
\usepackage[options]{package}   %引入宏包
...
\begin{document}                %正文
...
\end{document}
```

常用的文檔類（documentclass）有三種：`article`、`report`、`book`，它們的常用選項見表 2.1。

表 2.1: 文檔類常用選項

10pt, 11pt, 12pt	正文字號，預設10pt。LaTeX 會根據正文字號選擇標題、上下標等的字號。
letterpaper, a4paper	紙張尺寸，預設是 letter。
notitlepage, titlepage	標題後是否另起新頁。article 預設 notitlepage，report 和 book 預設有 titlepage。
onecolumn, twocolumn	欄數，預設單欄。
oneside, twoside	單面雙面。article 和 report 預設單面，book 預設雙面。
landscape	打印方向橫向，預設縱向。
openany, openright	此選項只用於 report 和 book。report 預設 openany，book 預設 openright。
draft	草稿模式。有時某些行排得過滿，draft 模式可以在它們右邊標上粗黑線提醒用戶。

LaTeX 的核心只提供基本的功能，系統以宏包（package）的形式提供附加功能或增強原有功能。其它一些編程語言也有類似的模塊化機制，比如 C/C++ 的 `#include`，Java 的 `import`。

2.4.2 標題、摘要、章節

一份文檔正文部分的開頭通常有標題、作者、摘要等信息，之後是章節等層次結構，內容則散佈於層次結構之間。

標題、作者、日期等命令如下，注意\maketitle 命令要放在最後。

```
\title{標題}
\author{作者}
\today
\maketitle
```

摘要環境用法如下：

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

常用的層次結構命令如下，

```
\chapter{...}
\section{...}
\subsection{...}
\subsubsection{...}
```

每個高級層次可以包含若干低級層次。article 中沒有 chapter，而 report 和 book 則支持上面所有層次。

2.4.3 目錄

我們可以用 \tableofcontents 命令來生成整個文檔的目錄，LaTeX 會自動設定目錄包含的章節層次，也可以用 \setcounter 命令來指定目錄層次深度。

```
\tableofcontents
\setcounter{tocdepth}{2}
```

如果不想讓某個章節標題出現在目錄中，可以使用以下帶 * 的命令來聲明章節。

```
\chapter*{...}
\section*{...}
\subsection*{...}
```

類似地，我們也可以用以下命令生成插圖和表格的目錄，插圖和表格功能將在後面章節中介紹。

```
\listoffigures
\listoftables
```

當章節或圖表等結構發生變化時，我們需要執行兩遍編譯命令以獲得正確結果。LaTeX 之所以設計成這樣可能是因爲當時的電腦內存容量有限。

2.5 文字排版

2.5.1 字符輸入

文檔中可以輸入的內容大致可以分爲：普通字符、控制符、特殊符號、注音符號、預定義字符串等。而這些內容有兩種輸入模式：文本模式（預設）和數學模式，普通的行間（inline）數學模式用`\$...\$`來表示。

LaTeX 中有些字符（例如 `# $ % ^ & _ { } ~ \` 等）被用作特殊的控制符，所以不能直接輸入，多數需要在前面加個 `\`。而 `\` 本身則要用 `\textbackslash` 命令來輸入，因爲 `\\` 被用作了換行指令。很奇怪爲什麼不用 C 語言的 `\n`，也許是因爲 TeX 的編程語言是 Pascal。

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

表 2.2 提供了一些符號的輸入方法示例，完整的符號列表見 Scott Pakin的《The Comprehensive LaTeX Symbol List》[1]。

2.5.2 換行、換頁、斷字

通常 LaTeX 會自動換行、換頁。用戶也可以用 `\\` 或 `\newline` 來強制換行；用 `\newpage` 來強制換頁。

一般情況下 LaTeX 會儘量均勻地斷字（Hyphenate），使得每一行的字間距分佈整齊。但有時我們也需要顯式指明斷字位置，比如下例就指明 BASIC 這個詞不能斷開，而 blar-blar-blar 可以在-處斷開。

```
\hyphenation{BASIC blar-blar-blar}
```

表 2.2: 一些符號和預定義字符串

特殊符號		注音符號		預定義字符串	
©	\textcopyright	å	\aa	September 25, 2008	\today
®	\textregistered	Å	\AA	TEX	\TeX
°C	$^\circ$C	æ	\ae	LATEX	\LaTeX
¥	\textyen	ø	\o	LATEX 2ε	\LaTeXe
£	\pounds	ö	\"o	METAFONT	\MF
€	\texteuro	ô	\^o	METAPOST	\MP
...	\dots	õ	\~o		

2.5.3 字樣、字號

LATEX 會自動調整正文、標題、章節、上下標、脚註等的字樣[3]、字號。我們也可以用表 2.3中的命令來設置字樣；用表 2.4中的命令來設置相對字號，比如正文字號是 10pt、11pt、12pt 時，tiny的字號就分別是 5pt、6pt、6pt。

LATEX 有一個特別的字樣強調命令：\emph，它在不同字樣和裝飾環境下有不同效果。比如周圍文字是正體，它就是斜體；反之它就是正體。

表 2.3: 字樣命令

\textrm{...}	roman	\textbf{...}	**bold face**
\textsf{...}	sans serif	\textit{...}	*italic*
\texttt{...}	typewriter	\textsl{...}	*slanted*
\emph{...}	*emphasized*	\underline{...}	underline
\textsc{...}	Small Caps		

[3]關於字樣詳見8.1節

表 2.4: 字號命令

	正文字號		
命令	10pt	11pt	12pt
\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

2.6 常用命令環境

2.6.1 列表

LaTeX 中有三種列表環境：itemize、enumerate、description，它們的一般用法如下：

```
\begin{itemize}
    \item C++
    \item Java
    \item HTML
\end{itemize}
```

- C++
- Java
- HTML

```
\begin{enumerate}
    \item C++
    \item Java
    \item HTML
\end{enumerate}
```

1. C++
2. Java
3. HTML

```
\begin{description}
    \item{C++} 一種編程語言
    \item{Java} 另一種編程語言
    \item{HTML} 一種標記語言
\end{description}
```

C++ 一種編程語言

Java 另一種編程語言

HTML 一種標記語言

2.6.2 對齊

LaTeX 中的段落預設兩端對齊（fully justified），我們也可以讓段落居左、居右或居中對齊。

```
\begin{flushleft}
本段落\\
居左
\end{flushleft}
```

本段落
居左

```
\begin{flushright}
本段落\\
居右
\end{flushright}
```

本段落
居右

```
\begin{center}
本段落\\
居中
\end{center}
```

本段落
居中

2.6.3 摘錄

LaTeX 中有三種摘錄環境：quote、quotation、verse。quote 兩端都縮進，quotation 在 quote 的基礎上增加了首行縮進，verse 比 quote 多了第二行起的縮進。

```
正文
\begin{quote}
引文兩端都縮進。
\end{quote}
正文
```

正文

引文兩端都縮進。

正文

```
正文
\begin{quotation}
引文兩端縮進，首行縮進。
\end{quotation}
正文
```

正文

引文兩端縮進，
首行縮進。

正文

```
正文
\begin{verse}
引文兩端縮進，第二行起縮進。
\end{verse}
正文
```

正文

引文兩端縮進，第
二行起縮進。

正文

2.6.4 原文照排

一般文檔中，命令和源代碼通常使用等寬字樣來表示，也就是原文照排。對此 LaTeX 提供了 `\verb` 命令（一般用於在正文中插入較短的命令）和 `verbatim` 環境。後者有帶 `*` 的版本用來標明空格。

```
正文中插入\verb|command|
\begin{verbatim}
printf("Hello, world!");
\end{verbatim}
\begin{verbatim*}
printf("Hello, world!");
\end{verbatim*}
```

正文中插入command

```
printf("Hello, world!");

printf("Hello, world!");
```

2.6.5 交叉引用

我們常常需要引用文檔中 `section`、`subsection`、`figure`、`table` 等對象的編號，這種功能叫作交叉引用 (cross referencing)。

LaTeX 中可以用 `\label{marker}` 命令來定義一個標記，標記名可以是任意字符串，但是在全文中須保持唯一。之後可以用 `\ref{marker}` 命令來引用標記處章節或圖表的編號，用 `\pageref{marker}` 來引用標記處的頁碼。

```
被引用處\label{sec}\\
...\\
第\pageref{sec}頁\ref{sec}節
```

被引用處
...
第19頁2.6.5節

文檔中新增交叉引用後，第一次執行 `latex` 或 `pdflatex` 編譯命令時會得到類似下面的警告信息。因爲第一次編譯只會掃瞄出有交叉引用的地方，第二次編譯才能得到正確結果。

```
LaTeX Warning: There were undefined references.
...
LaTeX Warning: Label(s) may have changed. Rerun to get cross-
references right.
```

2.6.6 脚註

脚註 (footnote) 的一般用法如下:

```
這裡是一段正文。\footnote{這裡是一段腳註。}
```

這裡是一段正文。[a]

[a] 這裡是一段脚註。

2.7 長度單位

LaTeX 中的常用長度單位如表 2.5 所示。point 是個傳統印刷業採用的單位，而 big point 是 Adobe 推出 PS 時新定義的單位。em 是個相對單位，比如當前字體是 11pt 時，1em 就是 11pt。

表 2.5: 常用長度單位

in	英吋	pt	point, 1/72.27 in	em	當前字體中字母M的寬度
cm	釐米	bp	big point, 1/72 in	ex	當前字體中字母X的高度
mm	毫米	pc	pica, 12 pt	mu	math unit，1/18 em

2.8 盒子

LaTeX 在排版時把每個對象（小到一個字母，大到一個段落）都視爲一個矩形盒子（box），我們在 HTML 和 CSS 中也可以見到類似的模型。

2.8.1 mbox 和 fbox

LaTeX 中最簡單的盒子是 `\mbox` 和 `\fbox`。前者把一組對象組合起來，後者在此基礎上加了個邊框。

```
\mbox{010 6278 5001}
\fbox{010 6278 5001}
```

010 6278 5001
010 6278 5001

2.8.2 makebox 和 framebox

稍複雜的 `\makebox` 和 `\framebox` 提供了寬度和對齊方式控制選項。這裡用 l、r、s 分別代表居左、居右和分散對齊。

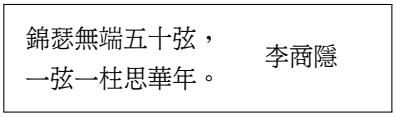
```
%語法：[寬度][對齊方式]{內容}
\makebox[100pt][l]{居左}
\framebox[100pt][r]{居右}
```

居左
居右

2.8.3 parbox 和 minipage

大一些的對象比如整個段落可以用 `\parbox` 命令和 `\minipage` 環境，兩者語法類似，也提供了對齊方式和寬度的選項。但是這裡的對齊方式是指與周圍內容的縱向關係，用 t、c、b 分別代表居頂、居中和居底對齊。

```
%語法：[對齊方式]{寬度}{內容}
\parbox[c]{90pt}{錦瑟無端五十弦，\\一弦一柱思華年。}李商隱
```



細心的讀者會發現 \parbox 和 \minipage 的選項排列順序和 \makebox 和 \framebox 的不一致，可能出自不同的作者。

參考文獻

[1] Scott Pakin. *The Comprehensive LaTeX Symbol List*, 2008. URL http://www.ctan.org/tex-archive/info/symbols/comprehensive/.

看什麼看，沒見過空白頁？

第三章　數學

> 今有上禾三秉，中禾二秉，下禾一秉，實三十九斗；上禾二秉，中禾三秉，下禾一秉，實三十四斗；上禾一秉，中禾二秉，下禾三秉，實二十六斗。問上、中、下禾實一秉各幾何？
>
> $$3x + 2y + z = 39$$
> $$2x + 3y + z = 34$$
> $$x + 2y + 3z = 26$$
>
> —《九章算術》

爲了使用 $\mathcal{AMS}$-LaTeX 提供的數學功能，我們首先需要在文檔的序言部分加載 `amsmath` 宏包，其詳細用法可參閱《amsmath User's Guide》[1]。更全面的數學內容排版可參閱 George Grätzer[1]的《More Math into LaTeX, 4th Edition》[2]。

3.1　數學模式

LaTeX 中的數學模式有兩種形式：inline 和 display。前者是指在正文插入行間數學公式，後者獨立排列，可以有或沒有編號。

行間公式用一對 `$...$` 來輸入，獨立公式用 `equation` 或 `equation*` 環境來輸入，有 `*` 的版本不生成公式編號。

前文提到 `\fbox` 命令可以給文本內容加個方框，數學模式下也有類似的命令 `\boxed`。

[1] 匈牙利裔，加拿大 Manitoba 大學數學系教授。

```
愛因斯坦的$E=mc^2$方程
\begin{equation}
    E=mc^2
\end{equation}
\[ E=mc^2 \]
\[ \boxed{E=mc^2} \]
```

愛因斯坦的$E = mc^2$方程

$$E = mc^2 \tag{3.1}$$

$$E = mc^2$$

$$\boxed{E = mc^2}$$

3.2 基本原素

3.2.1 字母

英文字母在數學模式下可以直接輸入，希臘字母則需要用表 3.1 中的命令輸入，注意大寫希臘字母的命令首字母也是大寫。

表 3.1: 希臘字母

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

3.2.2 指數、下標、根號

指數或上標用 ^ 表示，下標用 _ 表示，根號用 \sqrt 表示。上下標如果多於一個字母或符號，需要用一對 {} 括起來。

```
\[x_{ij}^2\quad \sqrt[2]{x}\]
```

$$x_{ij}^2 \quad \sqrt[2]{x}$$

3.2.3 分數

分數用 `\frac` 命令表示，它會自動調整字號，比如在行間公式中小一點，在獨立公式則大一點。`\dfrac` 命令把分數的字號顯式設置爲獨立公式中的大小，`\tfrac` 命令則把字號設爲行間公式中的大小。

```
$\frac{1}{2} \dfrac{1}{2}$
\[\frac{1}{2} \tfrac{1}{2}\]
```

$\frac{1}{2} \dfrac{1}{2}$

$$\frac{1}{2} \tfrac{1}{2}$$

3.2.4 運算符

有些小的運算符（operator）例如 `+ - * /` 等可以直接輸入，另一些則需要特殊命令。完整的數學符號參見 Scott Pakin 的《The Comprehensive LaTeX Symbol List》[3]。

```
\[\pm \times \div \cdot \cap \cup \geq \leq \neq \approx \equiv\]
```

$$\pm \times \div \cdot \cap \cup \geq \leq \neq \approx \equiv$$

和、積、極限、積分等大運算符用 `\sum \prod \lim \int` 等表示。它們的上下標在行間公式中被壓縮，以適應行高。。

```
$\sum_{i=1}^n i \prod_{i=1}^n \lim_{x\to0}x^2 \int_a^b x^2 dx\$
\[\sum_{i=1}^n i \prod_{i=1}^n \lim_{x\to0}x^2 \int_a^b x^2 dx\]
```

$\sum_{i=1}^n i \quad \prod_{i=1}^n \lim_{x\to0}x^2 \int_a^b x^2 dx$

$$\sum_{i=1}^n i \quad \prod_{i=1}^n \lim_{x\to0}x^2 \int_a^b x^2 dx$$

多重積分如果用多個 `\int` 來輸入的話，積分號間距過寬。正確的方法是用 `\iint \iiint \iiiint \idotsint` 等命令輸入。從下例中可以看出兩種方法的差異。

$$\int\int \quad \int\int\int \quad \int\int\int\int \quad \int\cdots\int$$
$$\iint \quad \iiint \quad \iiiint \quad \idotsint$$

3.2.5 分隔符

各種括號用 `() [] \{\} \langle\rangle` 等命令表示，注意花括號通常用來輸入命令和環境的參數，所以在數學公式中它們前面要加 `\`。因爲 LaTeX 中的 `|` 和`\|` 的應用過於隨意，amsamth 宏包推薦用 `\lvert\rvert` 和 `\lVert\rVert` 取而代之。

我們可以在上述分隔符前面加 `\big \Big \bigg \Bigg` 等命令來調整大小。LaTeX 原有的方法是在分隔符前面加 `\left \right` 來自動調整大小，但是效果不佳，所以 amsmath 不推薦用這種方法。

$$\Big(\Big(\Big((x+y)\Big)\Big)\Big) \quad \Big[\Big[\Big[[x+y]\Big]\Big]\Big] \quad \Big\{\Big\{\Big\{\{x+y\}\Big\}\Big\}\Big\}$$
$$\Big\langle\Big\langle\Big\langle\langle x+y\rangle\Big\rangle\Big\rangle\Big\rangle \quad \Big\|\Big\|\|x+y\|\Big\|\Big\| \quad \Big\|\Big\|\|x+y\|\Big\|\Big\|$$

3.2.6 箭頭

表 3.2 列出了部分箭頭的輸入方法。另外還有兩個命令生成的箭頭可以根據上下標自動調整長度。

```
\[\xleftarrow{x+y+z}\quad
\xrightarrow[x<y]{a*b*c}\]
```

$$\xleftarrow{x+y+z} \quad \xrightarrow[x<y]{a*b*c}$$

3.2.7 標註

表 3.3 列出一些短的注音符號（accent），表 3.4 則列出一些長的標註符號。

表 3.2: 箭頭

$\leftarrow$	\leftarrow	$\longleftarrow$	\longleftarrow
$\rightarrow$	\rightarrow	$\longrightarrow$	\longrightarrow
$\leftrightarrow$	\leftrightarrow	$\longleftrightarrow$	\longleftrightarrow
$\Leftarrow$	\Leftarrow	$\Longleftarrow$	\Longleftarrow
$\Rightarrow$	\Rightarrow	$\Longrightarrow$	\Longrightarrow
$\Leftrightarrow$	\Leftrightarrow	$\Longleftrightarrow$	\Longleftrightarrow

表 3.3: 數學注音符號

$\acute{x}$	\acute{x}	$\tilde{x}$	\tilde{x}	$\mathring{x}$	\mathring{x}
$\grave{x}$	\grave{x}	$\breve{x}$	\breve{x}	$\dot{x}$	\dot{x}
$\bar{x}$	\bar{x}	$\check{x}$	\check{x}	$\ddot{x}$	\ddot{x}
$\vec{x}$	\vec{x}	$\hat{x}$	\hat{x}	$\dddot{x}$	\dddot{x}

表 3.4: 長標註符號

$\overline{xxx}$	\overline{xxx}	$\overleftrightarrow{xxx}$	\overleftrightarrow{xxx}
$\underline{xxx}$	\underline{xxx}	$\underleftrightarrow{xxx}$	\underleftrightarrow{xxx}
$\overleftarrow{xxx}$	\overleftarrow{xxx}	$\overbrace{xxx}$	\overbrace{xxx}
$\underleftarrow{xxx}$	\underleftarrow{xxx}	$\underbrace{xxx}$	\underbrace{xxx}
$\overrightarrow{xxx}$	\overrightarrow{xxx}	$\widetilde{xxx}$	\widetilde{xxx}
$\underrightarrow{xxx}$	\underrightarrow{xxx}	$\widehat{xxx}$	\widehat{xxx}

3.2.8 省略號

省略號用 `\dots` `\cdots` `\vdots` `\ddots` 等命令表示，注意 `\cdots` 和 `\dots` 的差別。

$$\dots \quad \cdots \quad \vdots \quad \ddots$$

3.2.9 空白間距

在數學模式中，我們可以用表 3.5中的命令生成不同的間距，注意負間距命令 `\!` 可以用來減小間距。

表 3.5: 空白間距

`\,`	3/18 em	`\quad`	1 em
`\:`	4/18 em	`\qquad`	2 em
`\;`	5/18 em	`\!`	-3/18 em

3.3 矩陣和行列式

數學模式下可以用 `array` 環境來生成行列表。參數 `{ccc}` 用於設置每列的對齊方式，`l`、`c`、`r` 分別表示左中右；`\\` 和 `&` 用來分隔行和列。

```
\[\begin{array}{ccc}
x_1 & x_2 & \dots \\
x_3 & x_4 & \dots \\
\vdots & \vdots & \ddots \\
\end{array}\]
```

$$\begin{array}{ccc}
x_1 & x_2 & \dots \\
x_3 & x_4 & \dots \\
\vdots & \vdots & \ddots
\end{array}$$

`amsmath` 有幾個類似的環境：`pmatrix`、`bmatrix`、`Bmatrix`、`vmatrix` 和 `Vmatrix`，它們和 `array` 的主要區別是會在表兩端加上 () [] {} || |||| 等分隔符，其次這些環境沒有列對齊方式參數。

行間公式可以用 `smallmatrix` 環境來生成排列緊密的小矩陣。

3.4 多行公式

有時一個公式太長一行放不下，或幾個公式需要寫成一組，這時我們就要用到 amsmath 提供的幾個適合多行公式的環境。

3.4.1 長公式

對於多行不需要對齊的長公式，我們可以用 multiline 環境。

```
\begin{multline}
x=a+b+c+\\
d+e+f+g
\end{multline}
```

$$x = a + b + c + \\ d + e + f + g \quad (3.2)$$

需要對齊的長公式可以用 split 環境，它本身不能單獨使用，因此也稱作次環境，必須包含在 equation 或其它數學環境內。split 環境用 \\ 和 & 來分行和設置對齊位置。

```
\[ \begin{split}
x=&a+b+c+\\
  &d+e+f+g
\end{split} \]
```

$$x = a + b + c + \\ d + e + f + g$$

3.4.2 公式組

不需要對齊的公式組用 gather環境，需要對齊的用 align。

```
\begin{gather}
a=b+c+d\\
x=y+z
\end{gather}
```

$$a = b + c + d \quad (3.3)$$
$$x = y + z \quad (3.4)$$

```
\begin{align}
a&=b+c+d\\
x&=y+z
\end{align}
```

$$a = b + c + d \quad (3.5)$$
$$x = y + z \quad (3.6)$$

multine、gather、align 等環境都有帶 * 的版本，它們不生成公式編號。

有多種條件的公式組用 cases 次環境。

```
\[ y=\begin{cases}
-x & x<0\\
x & x\geq0
\end{cases} \]
```

$$y=\begin{cases}-x & x<0\\ x & x\geq 0\end{cases}$$

3.5 定理和證明

LaTeX 提供了一個 \newtheorem 命令來定義定理之類的環境，其語法如下。

```
\newtheorem{環境名}[編號延續]{顯示名}[編號層次]
```

在下例中，我們定義了四個環境：定義、定理、引理和推論，它們都在一個 section 內編號，而引理和推論會延續定理的編號。

```
\newtheorem{defination}{定義}[section]
\newtheorem{theorem}{定理}[section]
\newtheorem{lemma}[theorem]{引理}
\newtheorem{corollary}[theorem]{推論}
```

定義了上述環境之後，我們就可以像下面這樣使用它們。

```
\begin{defination}
Java是一種跨平台的編程語言。
\end{defination}
```

定義 **3.5.1.** *Java*是一種跨平台的編程語言。

```
\begin{theorem}
咖啡因會使人的大腦興奮。
\end{theorem}
```

定理 **3.5.1.** 咖啡因會使人的大腦興奮。

```
\begin{lemma}
茶和咖啡都會使人興奮。
\end{lemma}
```

引理 **3.5.2.** 茶和咖啡都會使人興奮。

```
\begin{corollary}
晚上喝咖啡會導致失眠。
\end{corollary}
```

推論 **3.5.3.** 晚上喝咖啡會導致失眠。

proof環境可以用來輸入下面這樣的證明，它會在證明結尾輸入一個 QED 符號[2]。

```
\begin{proof}[命題‘‘物質無限可分’’的證明]
一尺之棰，日取其半，萬世不竭。
\end{proof}
```

命題“物質無限可分”的證明. 一尺之棰，日取其半，萬世不竭。 □

3.6 數學字體

和文本模式類似，數學模式下可以用表 3.6 中的命令選擇不同字體，其中有些字體需要加載 amsfonts 宏包。

表 3.6: 數學字體

預設	\mathbf	\mathit	\mathsf	\mathcal	\mathbb
XNZRC	**XNZRC**	*XNZRC*	XNZRC	$\mathcal{XNZRC}$	$\mathbb{XNZRC}$

參考文獻

[1] AMS. *amsmath User's Guide*, 2002. URL http://www.ams.org/tex/amslatex.html.

[2] George Grätzer. *More Math into LaTeX*. Springer, 4th edition, 2007. URL http://www.amazon.com/exec/obidos/tg/detail/-/0387322892/.

[3] Scott Pakin. *The Comprehensive LaTeX Symbol List*, 2008. URL http://www.ctan.org/tex-archive/info/symbols/comprehensive/.

[2]拉丁語 quod erat demonstrandum 的縮寫。

看什麼看，沒見過空白頁？

第四章　插圖

> A picture says more than a thousand words.
>
> — Shakespeare

當年 Knuth 開發 TEX 時，GIF、JPEG、PNG、EPS 等圖形格式還沒有問世，所以 DVI 不能直接支持這些格式。但是高手就是高手，Knuth 在 TEX 上留了一個後門：`\special` 命令，讓後面的 Driver 決定怎樣處理圖形。

這和當年老毛把港澳台，老鄧把釣魚島都「留給後人解決」有異曲同工之妙。曾經有位出版社的編輯看上了包老師寫的一個程序，要包老師改改當作教學輔助軟件出版，但是包老師手頭沒有 DOS 中斷的資料沒辦法加鼠標操作。該編輯說：你把鼠標驅動打包在軟件裡，讓用戶自己琢磨是怎麼回事。

下面我們會在 4.1 節介紹一下 LATEX 所用圖形格式，4.2 節介紹怎樣插入已有的圖形，4.4–4.6 節討論怎樣製作向量圖形。

4.1　圖形格式

LATEX 支持點陣圖形格式 JPEG 和 PNG，也支持向量格式 EPS 和 PDF。對於示意圖，我們應該首選向量格式；包含大量自然色彩的圖像（比如照片）應該選 JPEG，人工點陣圖像應該選 PNG。

4.1.1　EPS

80 年代中後期，PS 風頭之勁一時無兩，人們自然會考慮把它作爲文檔中嵌入圖形的標準格式。然而 PS 實在太強大，人們擔心嵌入文檔的 PS 會搞破壞，

於是就產生了戴著手銬的 Encapsulated PostScript（EPS）。出於同樣的原因，人們也擔心嵌入 HTML 的 ActiveX、Java Applet、JavaScript 中混入惡意代碼，所以才會對它們也有所限制。

早年間 DVI 經常被轉換爲 PS，所以 EPS 就成了 LaTeX 的標準圖形格式。

4.1.2 Driver 們的口味

dvips

`dvips` 喜歡 PS，所以就愛屋及烏只支持嵌入 EPS。MiKTeX 看不慣這種壟斷行爲，就把 `dvips` 破解，添加了對 JPEG 和 PNG 的支持。如果你反對這種黑客破解行徑，只好找軟件把其它圖形格式轉換爲 EPS。

pdfLaTeX

pdfLaTeX 支持 JPEG、PNG和PDF，不支持 EPS。傳說pdfLaTeX 不支持 EPS 的原因是 PS 解釋器的版權問題。包老師認爲這種說法不可信，因爲 1997 年 Hàn The Thành 發佈 pdfTeX 時 PS 已經被 PDF 趕超，Adobe 與其保護 PS 還不如保護 PDF。

LaTeX 有兩個宏包 `epstopdf` 和 `pst-pdf` 可以實時地（on the fly）把 EPS 轉換爲 PDF[1]。然而前者有安全漏洞，後者用法繁瑣，用戶最好還是用其它軟件事先把 EPS 轉爲 PDF。

dvipdfm

`dvipdfm` 支持 JPEG、PNG、PDF，不支持 EPS，但是它可以實時地調用 Ghostscript 把 EPS 轉爲 PDF。所以從圖形格式支持的角度來講，`dvipdfm` 比 `dvips` 和 pdfLaTeX 都好。

那位同學說了，你這麼多廢話作甚，直接告訴我們用 `dvipdfm` 不就完了。然而你別忘了 `dvipdfm` 需要 DVI 作爲輸入，不幸的是用來生成 DVI 的 `latex` 對 JPEG 和 PNG 有意見。

[1]在這裡on the fly是指在後台處理，用戶不用操心。包老師不確定把它翻譯爲「實時」是否合適，因爲 real time 通常被翻譯爲實時。對於用戶無須干涉、知情的情況，有人說 user transparent，也有人說 black box，語言還眞奇妙。

綜上所述，這些 driver 都不能把圖形格式痛快地通吃，所以同學們別著急，且聽包老師慢慢忽悠怎樣轉換和處理圖形格式。

4.1.3 圖形格式轉換

注意把點陣圖形轉換爲向量圖形並不能提高圖形本身的質量，正所謂「garbage in, garbage out」。

JPEG 和 PNG 的範圍框

作爲中間格式的 DVI 不包含圖形本身，它只記錄圖形的尺寸和文件名，因爲具體的圖形處理由後面的 driver 負責。DVI 中圖形的尺寸來自它的範圍框（bounding box），而 `latex` 無法從點陣圖形文件中提取這一信息，所以我們需要以某種方式把範圍框信息告訴它。

一種方法是打開圖形文件，記下尺寸，插入圖形時加上相應的參數。

另一種方法用 `ebb` 程序生成一個含範圍框信息的文件。比如下例會生成 `graph.bb` 文件，有了它插入圖形時就不需要範圍框參數。注意有時此程序算出的範圍框不准，不知道是它的 bug 還是包老師的人品問題。

```
ebb graph.jpg
```

其它格式轉爲EPS

有很多程序都可以把點陣圖形轉換爲 EPS，比如 ImageMagick，以及 a2ping/sam2p、bmeps、jpeg2ps、sam2p 等。

PS 從 Level 2 開始才支持點陣圖形壓縮，所以在把其它格式轉爲 EPS 時應儘量使用 Level 2 或 3，否則輸出的 EPS 會很大。

下面是一個 ImageMagick 中 `convert` 程序的例子。

```
convert photo.jpg eps2:photo.eps
```

另外還有一種 PS 虛擬打印機的方法，優點是可以把幾乎所有文件「打印」成 EPS，缺點是輸出的是 PS Level 1，即使驅動程序提供了其它 Level 的選項。

1. 找一個 PS 打印機驅動程序。Windows 安裝盤附帶很多打印機驅動，其中帶 PS 字樣的就是 PS 驅動。包老師選的是「HP Color LaserJet 8550-PS」，Adobe 提供的 PS 驅動效果不太好。其它驅動安裝過程可能稍有不同。

2. 安裝時端口選「FILE」，或者後面打印時選擇「Print to File」。高級選項裡的 PS 選項選「Encapsulated PostScript (EPS)」。

3. 打開點陣圖形文件，打印到上面的虛擬打印機，輸出的文件就是 EPS，但是它沒有範圍框。

4. 用 GSview 打開上面生成的 EPS，不用理會沒範圍框的警告，Options 菜單裡選上「EPS Clip」，用 File 菜單的「PS to EPS」生成含範圍框的 EPS。

其它格式轉爲 **PDF**

LaTeX 附帶的 `epstopdf` 程序[2]可以把 EPS 轉爲 PDF。類似地我們也可以安裝一個 PDF 虛擬打印機，用它來把其它圖形文件轉爲 PDF。

4.2 插入圖形

4.2.1 插入命令

如今萬事俱備，只欠插入。上回講到哪兒來著？Yeah，Knuth 的後門 `\special`。

用低級命令 `\special` 來插入圖形很不爽，於是 LaTeX v2.09 增加了 `epsf` 和 `psfig` 宏包。之後 LaTeX 2_ε 推出了更好的 `graphics` 和 `graphicx` 宏包，這兩個宏包有個共同的命令：`\includegraphics`。`graphicx` 版本的語法更簡單，功能更強大，所以一般推薦用它。

插入圖形的具體命令如下，如果是點陣圖形需要加範圍框參數（左上角和右下角坐標）。

```
\includegraphics[bb=0 0 410 307]{photo.jpg}
```

[2]這個命令行程序和上面提到的 epstopdf 宏包是兩樣東西。

若想省略文件後綴，可在插入圖形前使用兩個命令。前者指定一個後綴列表，讓 LaTeX 自行查找；後者告訴 LaTeX 未知後綴的都是 EPS。

```
\DeclareGraphicsExtensions{.eps,.mps,.pdf,.jpg,.png}
\DeclareGraphicsRule{*}{eps}{*}{}
```

4.2.2 縮放、旋轉

表 4.1 和表 4.2 的選項可以用來縮放、旋轉和裁剪插圖。

表 4.1: includegraphics 命令的縮放和旋轉選項

scale	縮放比例
width	寬度
height	高度
totalheight	範圍框高度，旋轉時它不等於高度
keepaspectratio	如果不使用它而同時指定插圖的寬度、高度，長寬比可能會失調；使用它時長寬比不變，寬度、高度都不超過指定參數
angle	旋轉角度
origin	旋轉原點

表 4.2: includegraphics 命令的裁剪選項

viewport	可視區域的左上角和右下角坐標。
trim	左、下、右、上四邊裁剪的數值。
clip	是否眞正裁剪。預設爲false，不執行裁剪，多出的部分就那樣放著；設置爲true時執行裁剪。似乎是多此一舉。

若想深入瞭解 LaTeX 插入圖形的功能，請參考 Keith Reckdahl 的《Using Imported Graphics in LaTeX and pdfLaTeX》[1]（簡稱epslatex）。

4.2.3 figure環境

插圖通常需要佔據大塊空白，所以在文字處理軟件中用戶經常需要調整插圖的位置。LaTeX 有一個 `figure` 環境可以自動完成這樣的任務，這種自動調整位置的環境稱作浮動環境。

```
\begin{figure}[htbp]%位置選項
\centering
\includegraphics[bb=0 0 410 307,scale=.8]{photo}
\caption{10個月大的Anna}
\label{fig:anna}
\end{figure}
```

圖 4.1: 10個月大的Anna

上述代碼中，`[htbp]` 選項用來指定插圖排版的理想位置，這幾個字母分別代表 here、top、bottom、float page，也就是固定位置、頁頂、頁尾、單獨的浮

動頁。我們可以使用這幾個字母的任意組合，一般不推薦單獨使用 [h]，因爲那個位置也許很不合適，LaTeX 會很生氣。

\centering 用來使插圖居中，\caption 命令設置插圖標題，LaTeX 會自動給浮動環境的標題加上編號。注意 label 應放在caption 之後，否則引用時指向的是前一個插圖。

4.2.4 插入多幅圖形

並排擺放，共享標題

當我們需要兩幅圖片並排擺放，並共享標題時，可以在 figure 環境中使用兩個 \includegraphics 命令。

```
\begin{figure}[htbp]
\centering
\includegraphics{left}
\includegraphics{right}
\caption{反清復明}
\end{figure}
```

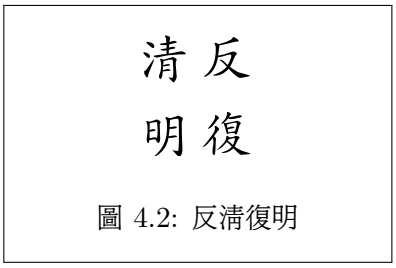


圖 4.2: 反清復明

並排擺放，各有標題

如果想要兩幅並排的圖片各有自己的標題，可以在 figure 環境中使用兩個 minipage 環境，每個環境裡插入一個圖。

```
\begin{figure}[htbp]
\centering
\begin{minipage}[t]{0.3\textwidth}
    \centering
    \includegraphics{left}
    \caption{清明}
\end{minipage}
```

```
\begin{minipage}[t]{0.3\textwidth}
    \centering
    \includegraphics{right}
    \caption{反覆}
\end{minipage}
\end{figure}
```


圖 4.3: 清明

圖 4.4: 反覆

並排擺放，共享標題，各有子標題

如果想要兩幅並排的圖片共享一個標題，並各有自己的子標題，可以使用 subfig 宏包提供的 \subfloat 命令。

subfloat 命令缺少寬度參數。雖然我們可以用 \hspace 命令調整子圖的距離，子標題卻只能和子圖本身一樣寬，就會出現折行。

```
\usepackage{subfig}
\begin{figure}[htbp]
\centering
\subfloat[清明]{
    \label{fig:subfig_a}
    \includegraphics{left}
}
\hspace{80pt}
\subfloat[反覆]{
    \label{fig:subfig_b}
    \includegraphics{right}
}
\caption{反清復明}
\end{figure}
```

每個子圖可以有各自的引用，就像這個樣子：圖 4.5a、圖 4.5b。

(a) 清明　(b) 反覆

圖 4.5: 反清復明

改進的子圖方法

爲了避免子標題折行，我們可以在 `\subfloat` 裡再嵌套個 `minipage`，因爲後者是有寬度的。

```
\begin{figure}[htbp]
\centering
\subfloat[清明]{
\label{fig:improved_subfig_a}
\begin{minipage}[t]{0.3\textwidth}
    \centering
    \includegraphics{left}
\end{minipage}
}
\subfloat[反覆]{
\label{fig:improved_subfig_b}
\begin{minipage}[t]{0.3\textwidth}
    \centering
    \includegraphics{right}
\end{minipage}
}
\caption{反清復明}
\end{figure}
```

清 反
明 復
(a) 清明 (b) 反覆

圖 4.6: 反清復明

4.3 圖形繪製工具比較

與 LaTeX 配套使用的繪圖工具主要有三種：METAPOST、PSTricks 和 PGF，它們的特點如下。

- 工作方式。METAPOST 離線繪圖，生成的 EPS 可以插入 LaTeX 文檔；PSTricks 和 PGF 都採用在線繪圖的方式，也就是 LaTeX 文檔內直接使用繪圖命令。
- 兼容性。METAPOST 生成的 MPS 需要先轉爲 PDF 才能被 pdfLaTeX 使用；PSTricks 生成的 EPS和 pdfLaTeX 不兼容；PGF 提供針對各種 driver 的接口，兼容性最好。
- 功能。PSTricks 有 PS 作後盾，功能最強；METAPOST 擅長處理數學內容；PGF 的流程圖有獨到之處。

限於篇幅，本文只對這三種工具進行簡介。除了它們，用戶也可以考慮一些面向 LaTeX 的繪圖前端，比如 Unix/Linux 下的 xfig 和 Windows 下的 TpX；或可以輸出 EPS 的專用軟件，比如 gnuplot 和 Matlab。

4.4 METAPOST

1989 年 John D. Hobby[3]開始設計一種繪圖語言及其編譯器，也就是 METAPOST。METAPOST 從 METAFONT 那裡獲得了大量靈感和源代碼，學生從導師

[3]Hobby 1985年從斯坦福獲博士學位，導師就是Knuth，現供職於貝爾實驗室。

那裡順點東西自然是手到擒來。METAPOST 和 METAFONT 語法類似，METAPOST 的主要優點在於是它輸出的是 EPS，而且支持彩色；METAFONT 輸出的是點陣格式，不支持彩色。Knuth 聲稱自己畫圖時只用 METAPOST。

從 Hobby 主頁上 METAPOST 的更新記錄看，它的最後版本是 0.63，年份是 1994。目前 Taco Hoekwater[4]繼續 METAPOST 的開發工作，最新版本是 1.005。

本文只對 METAPOST 作簡單介紹，若想深入瞭解請參閱 Hobby 的《A User's Manual for MetaPost》[2]。

4.4.1 準備工作

用戶一般需要把 METAPOST 源文件（.mp）用一個命令行程序 `mpost` 編譯爲一種特殊的 EPS，也稱作 MPS，然後再把 MPS 插入 LaTeX 源文件中使用。

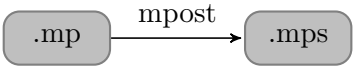


圖 4.7: MetaPost 的編譯

一個 METAPOST 源文件可以包含多個圖形，一般形式如下。代碼中每行語句以 ; 結尾，註釋行以 % 起始。每個圖形的繪圖命令包含在一對起始和結尾聲明之間。文件結尾也要有一個結尾聲明。

```
beginfig(1); %圖形起始
...          %繪圖命令
endfig;      %圖形結尾

beginfig(2);
...
endfig;
...
end;         %文件結尾
```

假如上面的源文件名字是 `fig.mp`，我們可以執行以下編譯命令。

[4]他也是LuaTeX開發者之一。

```
mpost fig(.mp)
```

編譯後就會生成「fig.1、fig.2、· · · 」等文件，每個文件的後綴就是相應的圖形起始聲明的編號。所以此編號在一個源文件中應保持唯一，否則後生成的文件就會覆蓋前面的。

這樣的文件名管理起來很麻煩，插入它們時也不能省略後綴，因爲 LaTeX 不能識別它們。用 \DeclareGraphicsExtensions 來逐一聲明後綴看起來很傻，自己改文件名更傻，\DeclareGraphicsRule 也顯得不夠嚴謹。

然而 METAPOST 已經考慮到這個問題，爲此提供了一個文件名模板命令。把下面的代碼加到源文件頭部，編譯輸出的文件名就會是「fig-01.mps、fig-02.mps、· · · 」。

```
filenametemplate "%j-%2c.mps";   %加在源文件頭部
```

我們也可以把這個命令加在每個圖形的起始聲明之前，指定個性化的輸出文件名，這樣可能更便於記憶。

```
filenametemplate "flowchart.mps" %加在每個圖形前面
```

MPS 可以用 GSview 查看，我們也可以用以下命令把它轉爲 PDF 再用 Adobe Reader 查看。

```
epstopdf flowchart.mps
```

4.4.2 基本圖形對象

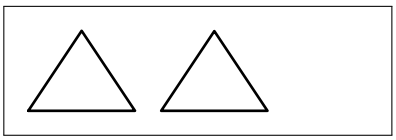
爲了節省空間，本節後面的示例會略去圖形起始聲明和結尾聲明等不重要的細節。

直線

繪圖命令 draw 把幾個點以直線段連接起來。METAPOST 中的預設長度單位是 bp，用戶也可以使用表 2.5 中的其它單位。我們還可以定義一個縮放係數，把坐標都轉換成此係數的倍數，這樣以後想縮放圖形時只要改這個係數即可。

注意 METAPOST 中的變量賦值符號是 :=，而 = 用於方程式。變量在同一源文件中只須定義一次，其後的圖形中都可以使用。

```
draw (0,0)--(40,0)--(20,20)--(0,0);
u:=10pt; %縮放係數
draw (5u,0)--(9u,0)--(7u,2u)--cycle;
```

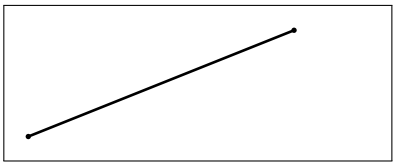


幾段直線或曲線可以構成一條路徑（path），在路徑末尾加個 `cycle` 就能構成封閉路徑（closed path）。上例中的兩個三角形看起來都是封閉的，但是前面這個其實不是眞正的封閉路徑。

4.4.3 點和線寬

`drawdot` 命令可以在指定坐標畫一個點，爲了使它醒目些我們可以換支粗一點的畫筆。METAPOST 中的畫筆預設是直徑 0.5pt 的圓形，拿它畫出來的線寬就是 0.5pt。

```
draw (0,0)--(10u,4u);
pickup pencircle scaled 2pt;
drawdot (0,0);
drawdot (10u,4u);
```



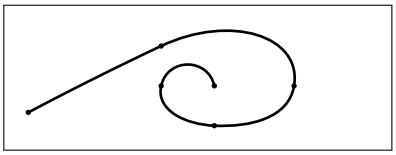
上面的 `pickup` 是一種全局操作，也就是說它會影響到之後所有的繪圖命令，我們也可以用 `withpen` 爲單個繪圖命令設置畫筆。

```
draw (0,0)--(10u,4u) withpen pencircle scaled 2pt;
```

曲線

曲線和直線的命令相近，只是把連接兩個點的 `--` 換成了 `..`。如果共用一些坐標，直線和曲線也可以混在一條語句裡畫。

```
draw (0,.5u)..(5u,3u)..(10u,1.5u)..
    (7u,0)..(5u,1.5u)..(7u,1.5u);
```



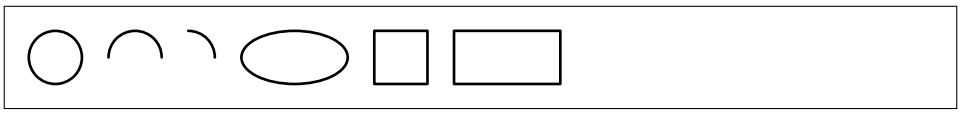
METAPOST 的曲線用三次貝茲（Cubic Bézier）算法實現。用戶可以在命令中增加 direction（方向）、Tension（張力）和 Curl（曲率）等控制，限於篇幅本文不贅述。

預定義圖形

fullcircle 命令以原點爲圓心畫一個單位圓，類似的預定義圖形還有 halfcircle、quartercircle、unitsquare 等。注意單位正方形的參考點在左下而不在其中心。

通過不同的橫向和縱向縮放係數，我們可以把圓形和正方形變成橢圓和長方形。

```
draw fullcircle scaled 2u;
draw halfcircle scaled 2u shifted (3u,0);
draw quartercircle scaled 2u shifted (5u,0);
draw fullcircle xscaled 4u yscaled 2u shifted (9u,0);
draw unitsquare scaled 2u shifted (12u,-u);
draw unitsquare xscaled 4u yscaled 2u shifted (15u,-u);
```

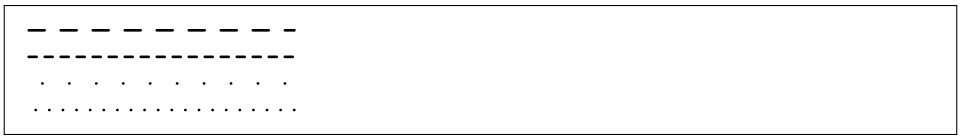


4.4.4 圖形控制

線型和箭頭

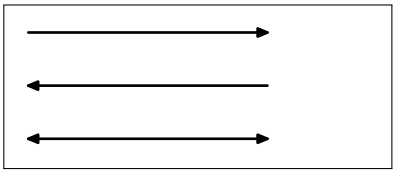
在繪製圖形時，我們不僅可以變換線寬，也可以使用多種線型。

```
draw (0,0)--(10u,0) dashed withdots;
draw (0,1u)--(10u,1u) dashed withdots scaled 2;
draw (0,2u)--(10u,2u) dashed evenly;
draw (0,3u)--(10u,3u) dashed evenly scaled 2;
```



箭頭和直線、曲線的語法相近，注意畫反向箭頭時需要把兩個坐標用一對 () 括起來。

```
drawarrow (0,4u)--(9u,4u);
drawarrow reverse ((0,2u)--(9u,2u));
drawdblarrow (0,0)--(9u,0);
```

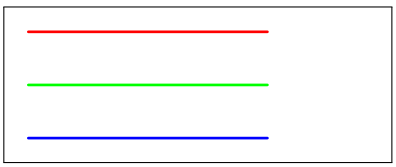


顏色和填充

METAPOST 預定義的顏色有黑、白、紅、綠、藍，它們的 RGB 值分別爲(0,0,0)、(1,1,1)、(1,0,0)、(0,1,0)、(0,0,1)，預設色就是黑色。

繪圖命令一般都可以通過 `withcolor` 參數來使用各種顏色。封閉路徑可以用 `fill` 命令填充。

```
draw (0,4u)--(9u,4u) withcolor red;
draw (0,2u)--(9u,2u) withcolor green;
draw (0,0)--(9u,0) withcolor blue;
```



```
fill p scaled u;
fill p scaled u shifted (3u,0) withcolor red;
fill p scaled u shifted (6u,0) withcolor green;
fill p scaled u shifted (9u,0) withcolor blue;
```

另一個命令 `filldraw` 可以看作是 `fill+draw`，它除了填充外還會把路徑用指定的畫筆畫一遍。然而不幸的是畫邊緣和填充內部只能用同一種顏色，所以它的用處不大。

除了爲每個繪圖命令單獨指定顏色，我們也可以使用一個全局命令，使得其後的繪圖命令都使用某種顏色。

```
drawoption(withcolor blue);
```

下面的方法可以用來定義基本色以外的顏色，用基本色混色或直接用RGB值效果是一樣的。

```
color c[];
c1 := .9red + .6green + .3blue;
c2 := (.9,.6,.3);
```

圖形變換

我們可以對路徑進行縮放、平移、旋轉等變換操作，橫向和縱向縮放可以分開進行。由於旋轉是圍繞原點進行的，所以要注意平移和旋轉的順序。下例中定義了一個 path 變量，以便後面重用。

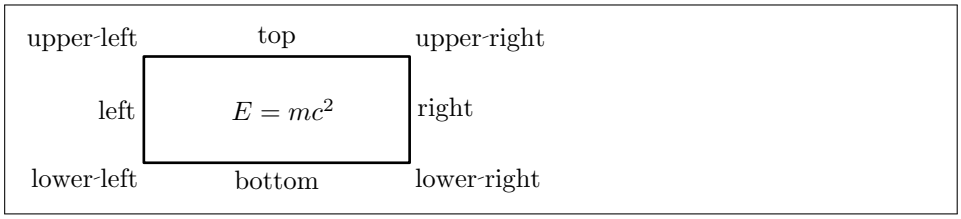
```
path p;
p := (0,0)--(2,0)--(1,1.732)--cycle;
draw p scaled u;
draw p xscaled 2u yscaled u shifted (3u,0);
draw p scaled u rotated 60 shifted (8u,0);
```

標註

\label 命令可以在指定的點附近加文字標註。METAPOST 也可以用一對 btex 和 etex 來嵌入一些 TEX 內容，比如數學標註。

```
draw unitsquare xscaled 10u yscaled 4u;
label.top("top", (5u,4u));
label.bot("bottom", (5u,0));
label.lft("left", (0,2u));
label.rt("right", (10u,2u));
```

```
label.ulft("upper left", (0,4u));
label.urt("upper right", (10u,4u));
label.llft("lower left", (0,0));
label.lrt("lower right", (10u,0));
label.rt(btex $E=mc^2$ etex, (3u,2u));
```



因爲用預設方法編譯生成的 MPS 不嵌入字體，當 METAPOST 包含文字時，GSview 就不能正常查看。這時我們可以給編譯命令加個參數，生成的 MPS 就會包含字體信息。注意這種方法生成的 MPS 雖然 GSview 能查看，`dvipdfmx` 卻不能正常處理。

```
mpost \prologues:=2; input fig.mp
```

4.4.5 編程功能

數據類型和變量

METAPOST 中有10 種基本數據類型：numeric、pair、path、pen、 color 、cmykcolor、transform、string、boolean、picture。我們已經接觸過其中幾種，比如縮放係數 u 就是一個 numeric，一個點的坐標是一個 pair，幾個點用直線或曲線連起來是一個 path，pencircle 是一種 pen，black 是一種 color，scaled、rotated、shifted 都是 transform。

numeric 類型變量的精度是 1/65536，它的絕對值不能超過 4096，在計算過程中數值可以達到 32768。這樣的規定也應歸功於當年的電腦硬件，不過對於科技文檔插圖而言，4096 一般還是夠用的。

除了預設的 numeric，其它變量在使用之前都需要用數據類型來顯式聲明。相同類型的變量可以在一行語句中聲明，但是帶下標的變量不能放在同一行（這個規定很蹊蹺）。

```
numeric x,y,z;    %正確
numeric x1,x2,x3; %錯誤
numeric x[];      %正確
```

數學運算

METAPOST 中可以使用普通的運算符，比如 `+ - * /`；也提供一些特殊的運算符，比如 `a++b` 表示$\sqrt{a^2+b^2}$，`a+-+b` 表示$\sqrt{a^2-b^2}$；另外表 4.3 列出一些常用數學函數。

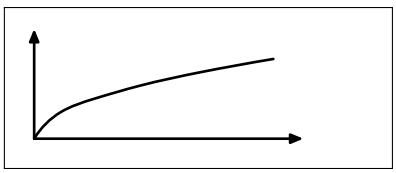
表 4.3: 數學函數

abs	絕對值	mexp	指數
round	四捨五入	mlog	對數
ceiling	向上圓整	sind	正弦
floor	向下圓整	cosd	餘弦
mod	模余	normaldeviate	正態分佈隨機數
sqrt	開方	uniformdeviate	均勻分佈隨機數

循環

當執行重複任務時，循環語句可以讓程序變得簡潔。注意下例中的循環語句是一條命令，之所以分成三行寫是爲了看起來清晰點。

```
draw (0,0) %注意這裡沒有分號
for x=1 upto 3:
    ..(x*x,x)*u
endfor;
```



循環語句預設步長是 1，我們也可以改用其它步長。`upto` 其實就是 `step 1 until` 的縮寫方式。

```
for x=1 step .5 until 3:
```

4.5 PSTricks

PSTricks 是一個基於 PS 的宏包，有了它用戶就可以直接在 LaTeX 文檔中插入繪圖命令。PSTricks 早期的作者是 Timothy Van Zandt[5]，初始開發年月不

[5]法國Insead大學經濟系教授。

詳，他於 1997 年退居二線，之後由 Denis Girou、Sebastian Rahtz[6]、Herbert Voß 等維護。

本文只介紹 PSTricks 的基本功能，若想深入瞭解請參閱 Van Zandt 的《PSTricks User's Guide》[3]。另外表 4.4 列出了一些可以和 PsTricks 配合使用的輔助宏包。

表 4.4: PSTricks 輔助宏包

multido	循環語句	pst-eucl	幾何函數
pst-plot	函數繪圖	pst-math	弧度三角函數
pst-plot3d	三維繪圖	pstricks-add	極坐標

4.5.1 準備工作

首先要引入 PSTricks 宏包。PSTricks 中預設長度單位是 1cm，我們也可以設置自己的單位。

```
\usepackage{pstricks}
\psset{unit=10pt}
```

繪圖命令一般要放在 `pspicture` 環境裡，這樣 LaTeX 就會給圖形預留一個矩形區域，注意這個矩形要能容納所有圖形對象。爲了節省空間，在本節後面的示例代碼中， `pspicture` 環境將被略去。

```
\begin{pspicture}(0,0)(4,2)
...
\end{pspicture}
```

另外需要注意的是，嵌入 LaTeX 的 PSTricks 生成的是 PS，`dvips` 可以處理，`dvipdfm` 和pdfLaTeX 則不能。所以如果使用後兩種 driver，需要先行生成 EPS。

第一種方法是把 PSTricks 代碼放進一個空白的LaTeX 頁面，用它生成一個簡單的 DVI。

[6]牛津大學計算機系網管。

```
\documentclass{article}
\usepackage{pstricks}
\pagestyle{empty}   %頁面樣式為空

\begin{document}
\psset{unit=10pt}
\colorbox{white}{%
    \begin{pspicture}(0,0)(4,2)%
    \psdot(0,0)%
    \psdots(0,2)(2,2)(4,2)%
    \end{pspicture}%
}
\end{document}
```

然後用以下命令把 DVI 轉爲 EPS，-E 參數即代表 EPS。上面代碼中的 colorbox 使得生成的 EPS 有正確的範圍框。

```
dvips pst_dots(.dvi) -E -o dots.eps
```

第二種方法是用 pst-eps 宏包，它能夠在線處理 PSTricks 代碼並生成 EPS，這樣用戶就可以在同一文件中使用該 EPS。

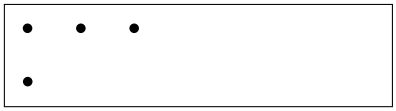
然而 dvipdfmx 不能正確處理 pst-eps 生成的 EPS。它和 \rput、\uput 命令，pst-plot 宏包中的 \psaxes 命令都不兼容。類似的 ps4pdf 宏包和 tabularx 宏包不兼容。

4.5.2 基本圖形對象

點

我們可以用以下命令畫一個或多個點。

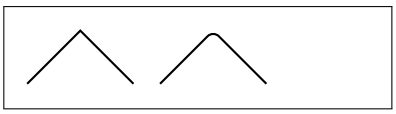
```
\psdot(0,0)
\psdots(0,2)(2,2)(4,2)
```



直線、多邊形、矩形

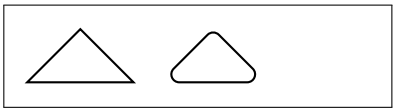
\psline 命令把多個點用直線段連接起來，線段之間的連接預設爲尖角，也可以設置圓角。

```
\psline(0,0)(2,2)(4,0)
\psline[linearc=.3](5,0)(7,2)(9,0)
```



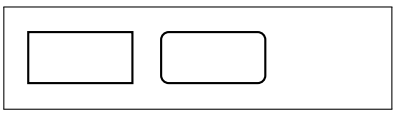
\pspolygon 命令和 \psline 類似，但是它會形成封閉路徑。

```
\pspolygon(0,0)(2,2)(4,0)
\pspolygon[linearc=.3](5,0)(7,2)(9,0)
```



矩形用 \psframe 命令，其參數就是矩形左下角和右上角的坐標。矩形也可以設置圓角。

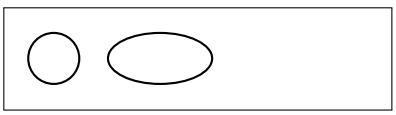
```
\psframe(0,0)(4,2)
\psframe[framearc=.3](5,0)(9,2)
```



圓、橢圓、圓弧、扇形

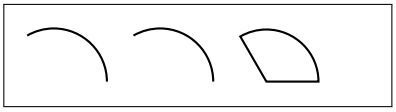
圓形用 \pscircle 命令，參數是圓心和半徑。橢圓用 \psellipse 命令，參數是中心、長徑、短徑。注意這兩個命令的半徑參數用不同的括號，可能是作者的筆誤。

```
\pscircle(1,1){1}
\psellipse(5,1)(2,1)
```



圓弧用 \psarc 命令，其參數是圓心、半徑、起止角度，逆時針作圖。\psarcn 類似，只是順時針作圖。扇形用 \pswedge 命令。

```
\psarc(1,0){2}{0}{120}
\psarcn(5,0){2}{120}{0}
\pswedge(9,0){2}{0}{120}
```

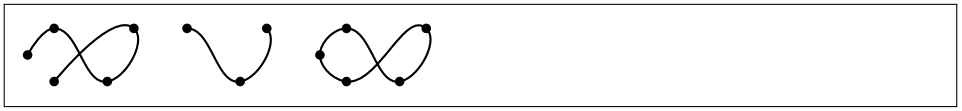


曲線

\pscurve 命令把一系列點用平滑曲線連接起來；它的變形版本 \psecurve 命令不顯示曲線的兩個端點；另一變形命令 \psccurve 則把曲線封閉起來。

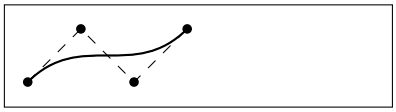
參數 showpoints=true 用來顯示曲線的構成點，此參數也可用於其它繪圖命令。

```
\pscurve[showpoints=true](0,1)(1,2)(3,0)(4,2)(1,0)
\psecurve[showpoints=true](5,1)(6,2)(8,0)(9,2)(5,0)
\psccurve[showpoints=true](11,1)(12,2)(14,0)(15,2)(12,0)
```



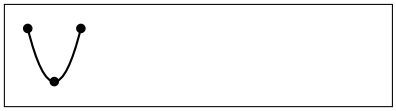
\psbezier 命令輸出一條貝茲曲線，其參數就是曲線的控制點。

```
\psbezier[showpoints=true]
    (0,0)(2,2)(4,0)(6,2)
```



拋物線用 \psparabola 命令，它有兩個參數，第一個是拋物線通過的一點，第二個是拋物線的頂點。

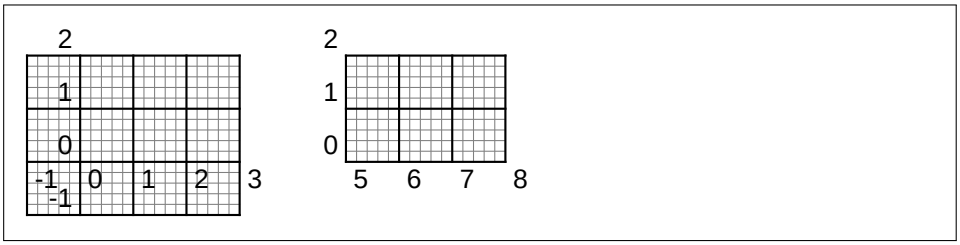
```
\psparabola[showpoints=true]
    (2,2)(1,0)
```



網格和坐標

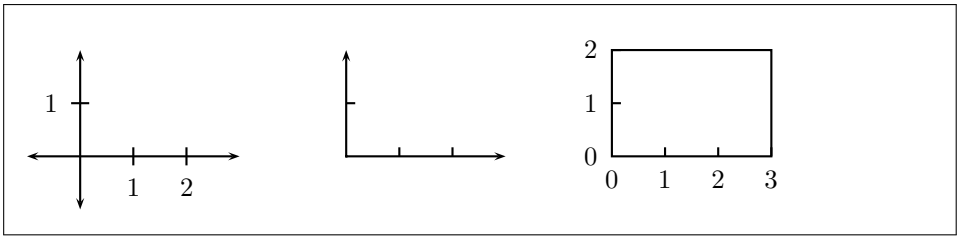
科技製圖通常會用到坐標網格。\psgrid 命令輸出一個矩形網格，它有三個參數點。網格坐標標註在通過第一個點的兩條直線上，第二和第三個點是矩形的兩個對角頂點。當第一個參數省略時，坐標標註在通過第一個頂點的兩條矩形邊上。

```
\psgrid(0,0)(-1,-1)(3,2)
\psgrid(5,0)(8,2)
```



pst-plot 宏包提供的 \psaxes 命令輸出坐標軸。它的參數和 \psgrid 的類似，刻度和標註都可以靈活地設置，也可以把坐標軸改成一個矩形框的形式。

```
\psset{unit=10pt}
\psaxes{<->}(0,0)(-1,-1)(3,2)
\psaxes[tickstyle=top,labels=none]{->}(5,0)(8,2)
\psaxes[axesstyle=frame,tickstyle=top]{->}(10,0)(13,2)
```

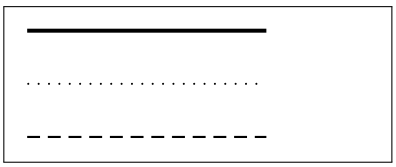


4.5.3 圖形控制

線型和箭頭

PSTricks 中的預設線寬是 0.8pt，預設線型是實線。以下參數可以控制單個繪圖命令的線寬和線型。

```
\psline[linewidth=1.5pt](0,4)(9,4)
\psline[linestyle=dotted](0,2)(9,2)
\psline[linestyle=dashed](0,0)(9,0)
```

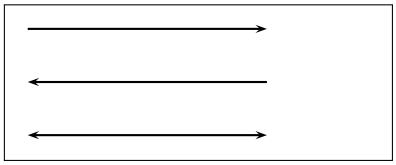


我們也可以用命令 \psset 命令來設置全局參數。

```
\psset{linewidth=1pt,linestyle=dashed}
```

以下參數可以控制繪圖命令的箭頭。

```
\psline{->}(0,4)(9,4)
\psline{<-}(0,2)(9,2)
\psline{<->}(0,0)(9,0)
```



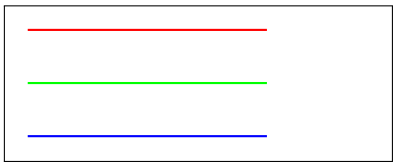
顏色和填充

PSTricks 預定義的顏色有 black、darkgray、gray、lightgray、white 等灰度顏色，也有 red、green、blue、cyan、magenta、yellow 等彩色。我們也可以自定義灰度顏色和彩色。

```
\newgray{mygray}{.3}
\newrgbcolor{mycolor}{.3 .4 .5}
```

以下參數可以控制單個繪圖命令的顏色，我們也可以用 `\psset` 命令設置全局參數。

```
\psline[linecolor=red](0,4)(9,4)
\psline[linecolor=green](0,2)(9,2)
\psline[linecolor=blue](0,0)(9,0)
```



以下參數可以控制單個繪圖命令的填充模式和填充顏色，注意只有封閉路徑才可以填充。

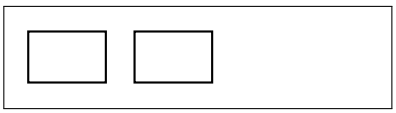
```
\pscircle[fillstyle=solid,fillcolor=red](1,1){1}
\pscircle[fillstyle=vlines](4,1){1}
\pscircle[fillstyle=hlines](7,1){1}
\pscircle[fillstyle=crosshatch](10,1){1}
```

4.5.4 對象佈局

平移

參數 origin 可以讓一個圖形對象平移到指定的坐標點，我們也可以用 \psset 命令設置全局平移參數。

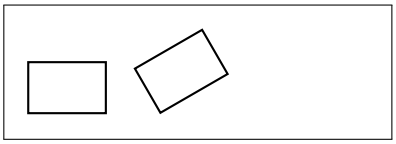
```
\psframe(0,0)(3,2)
\psframe[origin={4,0}](0,0)(3,2)
```



旋轉

\rput 命令可以對一個圖形對象同時進行旋轉和平移操作。它有兩個參數，第一個是旋轉角度，第二個是平移到的坐標點。

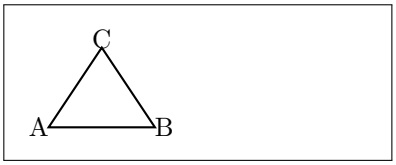
```
\psframe(0,0)(3,2)
\rput{30}(5,0){\psframe(0,0)(3,2)}
```



文字標註

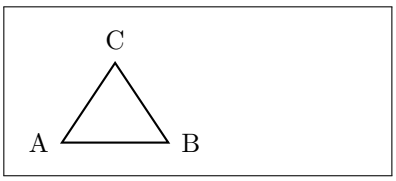
\rput 命令還可以在指定的坐標點標註文字，這時它的第一個參數是坐標點相對於標註的位置。其取值可以是縱向的 t、b（上下），或橫向的 l、r（左右），也可以是縱向和橫向位置的組合。

```
\pspolygon(0,0)(4,0)(2,3)
\rput[r](0,0){A}
\rput[l](4,0){B}
\rput[b](2,3){C}
```



\rput 生成的標註就在坐標點上，有時會感覺離圖形太近。另一個命令 \uput 則生成預設距離指定坐標點 5pt的標註，它的第一個參數是標註相對於坐標點的角度。

```
\pspolygon(0,0)(4,0)(2,3)
\uput[l](0,0){A}
\uput[r](4,0){B}
\uput[u](2,3){C}
```



\uput 的角度參數可以是任意角度，也可以是字母，參見。注意 \uput 的角度參數和 \rput 命令的參考點位置參數的定義幾乎正好相反，這也許反映了作者灑脫的風格。

表 4.5: uput 命令的角度參數

r	0°	ur	45°
u	90°	ul	135°
l	180°	dl	225°
d	270°	dr	315°

4.6 PGF

PGF 和 Beamer 的作者都是 Till Tantau[7]。Tantau 當初開發 Beamer 是爲了應付 2003 年他的博士學位論文答辯，之後它在 CTAN 上流行開來。2005 年 PGF 從 Beamer 項目中分離出來，成爲一個獨立的宏包。

本文只對 PGF 作簡單介紹，若想深入瞭解請參閱 Tantau 的《TikZ and PGF Manual》[4]。

4.6.1 準備工作

一般人們並不直接使用 PGF 命令，而是通過它前端 TikZ 來調用 PGF。在引用 tikz 宏包之前，用戶需要設置 PGF 系統 driver。

```
\def\pgfsysdriver{pgfsys-dvipdfmx.def}
\usepackage{tikz}
```

PGF 的預設長度單位是 1cm，我們也可以改用其它單位。注意這樣預定義的長度單位有時會失效，這可能是 PGF 的 bug。

```
\pgfsetxvec{\pgfpoint{10pt}{0}}
\pgfsetyvec{\pgfpoint{0}{10pt}}
```

[7]德國Lübeck大學計算機研究所教授。

TikZ 提供一個 tikz 命令和一個 tikzpicture 環境，具體繪圖指令可以放在 tikz 後面，也可以放在 tikzpicture 中間。兩種方法效果相同，用戶可以任意選擇。爲了節省空間，本節的示例將省略部分環境代碼。

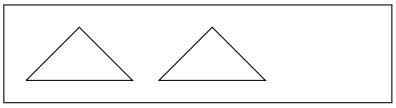
```
\tikz ...   %繪圖命令
\begin{tikzpicture}
...         %繪圖命令
\end{tikzpicture}
```

4.6.2 基本圖形對象

直線、多邊形、矩形

TikZ 中直線的語法和 METAPOST 類似，加 cycle 參數才能構成眞正的封閉路徑。

```
\draw (0,0)--(4,0)--(2,3)--(0,0);
\draw (5,0)--(9,0)--(7,3)--cycle;
```



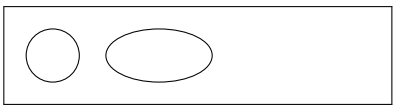
矩形命令如下，它的兩個參數是矩形的兩個對角頂點。

```
\draw (0,0) rectangle (4,2);
```

圓、橢圓、弧

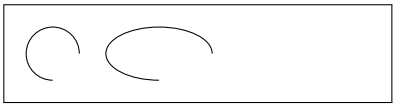
圓和橢圓命令如下，圓的參數是圓心和半徑，橢圓的參數是中心、長徑、短徑。

```
\draw (1,1) circle (1);
\draw (5,1) ellipse (2 and 1);
```



圓弧和橢圓弧命令如下，圓弧的參數是起始點，起始角度、終止角度、半徑；橢圓弧則把半徑換成了長徑和短徑。

```
\draw (2,1) arc (0:270:1);
\draw (7,1) arc (0:270:2 and 1);
```



曲線和拋物線

曲線命令如下，中間的參數是控制點。

```
\draw (0,0) .. controls (2,2)
    and (4,2) .. (4,0);
```

拋物線命令如下，除了起止點還可以指定頂點。

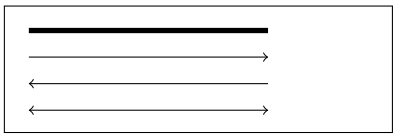
```
\draw (-1,1) parabola
    bend (0,0) (1.414,2);
```

4.6.3 圖形控制

線型和箭頭

繪圖命令可以設置線型和箭頭參數。

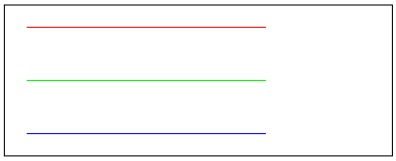
```
\draw[line width=2pt] (0,0)--(9,0);
\draw[->] (0,1)--(9,1);
\draw[<-] (0,2)--(9,2);
\draw[<->] (0,3)--(9,3);
```



顏色、填充、陰影

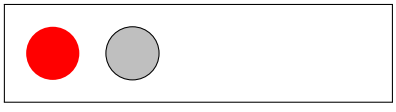
顏色參數的用法如下。PGF 可以使用 `xcolor` 宏包[5]中定義的所有顏色。

```
\draw[red] (0,4)--(9,4);
\draw[green] (0,2)--(9,2);
\draw[blue] (0,0)--(9,0);
```



封閉路徑可以用顏色填充，`\filldraw` 命令可以分別指定邊框色和填充色。

```
\fill[red] (1,1) circle (1);
\filldraw[fill=lightgray,draw=black]
    (4,1) circle (1);
```



\shade 命令可以產生漸變和光影效果，預設是從上到下，灰色漸變爲白色。我們也可以使用其它方向和顏色的漸變。

```
\shade (0,0) rectangle (2,2);
\shade[left color=red,right color=orange] (3,0) rectangle (5,2);
\shade[inner color=red,outer color=orange] (6,0) rectangle (8,2);
\shade[ball color=blue] (10,1) circle (1);
```

圖形變換

對圖形對象可以進行平移和旋轉操作，注意如果兩種操作同時進行，它們是有順序的。注意預定義的長度單位在這裡對平移參數失效。

```
\draw (0,0) rectangle (2,2);
\draw[xshift=30pt] (0,0) rectangle (2,2);
\draw[xshift=75pt,rotate=45] (0,0) rectangle (2,2);
```

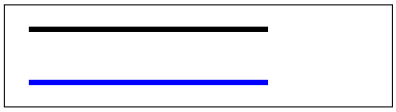
4.6.4 樣式

PGF 比 METAPOST 和 PSTricks 多了一個有趣的概念：樣式 (style) ，它的思路和 HTML 的 CSS 相近。我們可以先定義兩種樣式，

```
\tikzset{
    myline/.style={line width=2pt},
    myblueline/.style={myline,blue}
}
```

然後就可以在繪圖命令中這樣使用樣式。

```
\draw[myline] (0,2)--(9,2);
\draw[myblueline] (0,0)--(9,0);
```



除了用 \tikzset 命令定義樣式，我們也可以在 tikzpicture 環境頭部聲明樣式。前者是全局性的，後者則是局部性的。

```
\begin{tikzpicture}[
    thickline/.style=2pt,
    bluethickline/.style={thickline,color=blue}
]
...
\end{tikzpicture}
```

注意在樣式中預定義長度單位有時會失效，所以最好使用絕對單位。

4.6.5 流程圖

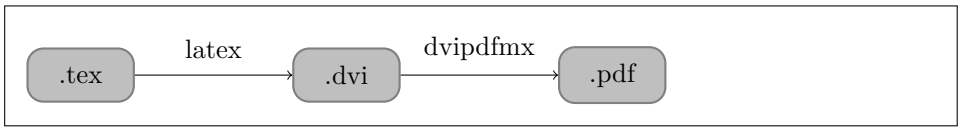
節點

PGF 中的節點（node）可以是簡單的標籤，也可以有各種形狀的邊框，還可以有各種複雜的屬性。比如下例中的節點樣式：box，它的邊框是矩形，有圓角；它有最小寬度、高度、文字和邊框的距離，邊框和填充顏色等屬性。

```
\tikzset{
    box/.style={rectangle, rounded corners=6pt,
        minimum width=50pt, minimum height=20pt, inner sep=6pt,
        draw=gray,thick, fill=lightgray}
}
```

除了上述類別屬性，節點還可以有名字、位置等屬性。在下例中，我們先畫了三個有名字的文本框；然後用箭頭把文本框連接起來，注意連接時要引用文本框的名字；接著在箭頭上加了標籤。

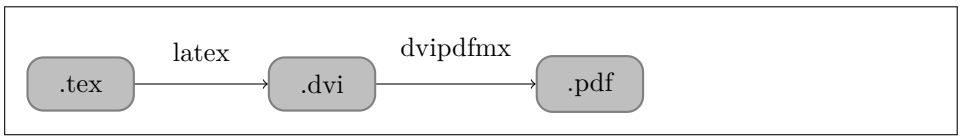
```
\node[box] (tex) at(0,0) {.tex};  %文本框
\node[box] (dvi) at(10,0) {.dvi}; %文本框
\node[box] (pdf) at(20,0) {.pdf}; %文本框
\draw[->] (tex)--(dvi);           %箭頭
\draw[->] (dvi)--(pdf);           %箭頭
\node at (5,1) {latex};           %標籤
\node at (15,1) {dvipdfmx};       %標籤
```



在上例中的節點都使用了絕對位置，PGF 中還可以使用更靈活一點的相對位置。比如在下例中，dvi 節點在 tex 節點右邊 50pt 處（我們前面定義的基本長度單位是 10pt），而 pdf 節點又在 dvi 節點右邊 50pt 處。

箭頭可以換爲專門用來連接節點的 `edge` ；標籤也改成相對位置，箭頭上方 5pt 處。

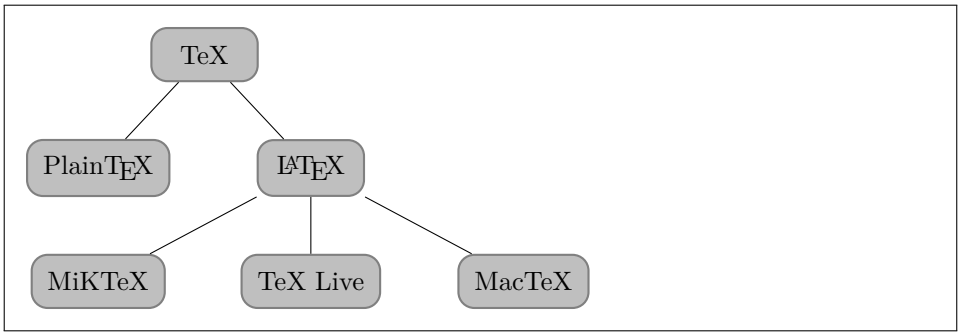
```
\node[box] (tex) {.tex};
\node[box,right=5 of tex] (dvi) {.dvi};
\node[box,right=6 of dvi] (pdf) {.pdf};
\path (tex) edge[->]  node[above=.5] {latex} (dvi)
    (dvi) edge[->] node[above=.5] {dvipdfmx} (pdf);
```



樹

下面是一棵簡單的樹。我們可以用一個參數控制相鄰節點的距離，預定義長度單位對此參數也會失效。

```
\begin{tikzpicture}[sibling distance=80pt]
\node[box] {TeX}
    child {node[box] {Plain\TeX}}
    child {node[box] {\LaTeX}
        child {node[box] {MiKTeX}}
        child {node[box] {TeX Live}}
        child {node[box] {MacTeX}}
    };
\end{tikzpicture}
```



參考文獻

[1] Keith Reckdahl. *Using Imported Graphics in LaTeX and pdfLaTeX*, 2006. URL http://www.ctan.org/tex-archive/info/epslatex/english/.

[2] John D. Hobby. *MetaPost: A User's Manual*, 2007. URL http://www.ctan.org/tex-archive/graphics/metapost/.

[3] Timothy van Zandt. *PSTricks User's Guide*, 2007. URL http://www.ctan.org/tex-archive/graphics/pstricks/base/doc/.

[4] Till Tantau. *TikZ and PGF Manual*, 2008. URL http://sourceforge.net/projects/pgf/.

[5] Uwe Kern. *Extending LaTeX's Color Facilities: The xcolor Package*. CTAN, 2007. URL http://www.ukern.de/tex/xcolor.html.

第五章　表格

5.1　簡單表格

tabular 環境提供了最簡單的表格功能。它用 \hline 命令代表橫線，| 代表豎線，用 & 來分欄。每個欄位的對齊方式可以用 l、c、r（左中右）來控制。

```
\begin{tabular}{|l|c|r|}
    \hline
    作業系統 & 發行版 & 編輯器 \\
    \hline
    Windows & MikTeX & TeXnicCenter \\
    \hline
    Unix/Linux & TeX Live & Emacs \\
    \hline
    Mac OS & MacTeX & TeXShop \\
    \hline
\end{tabular}
```

作業系統	發行版	編輯器
Windows	MikTeX	TeXnicCenter
Unix/Linux	TeX Live	Emacs
Mac OS	MacTeX	TeXShop

和針對插圖的 figure 環境類似，LaTeX 還有另一個針對錶格的浮動環境 table。我們可以用它給上面的示例穿件馬甲，順便把表格簡化爲科技文獻中

常用的三線表。

```
\begin{table}[htbp]
\caption{浮動環境中的三線表}
\label{tab:threesome}
\centering
\begin{tabular}{lll}
    \hline
    作業系統 & 發行版 & 編輯器 \\
    \hline
    Windows & MikTeX & TeXnicCenter \\
    Unix/Linux & TeX Live & Emacs \\
    Mac OS & MacTeX & TeXShop \\
    \hline
\end{tabular}
\end{table}
```

表 5.1: 浮動環境中的三線表

作業系統	發行版	編輯器
Windows	MikTeX	TeXnicCenter
Unix/Linux	TeX Live	Emacs
Mac OS	MacTeX	TeXShop

完美主義者可能覺得上面示例中的三條線一樣粗不夠美觀，這時可以使用 `booktabs` 宏包[1]的幾個命令。

```
\begin{table}[htbp]
\caption{浮動環境中的三線表}
\centering
\begin{tabular}{lll}
    \toprule
    作業系統 & 發行版 & 編輯器 \\
    \midrule
    Windows & MikTeX & TeXnicCenter \\
    Unix/Linux & TeX Live & Emacs \\
```

```
    Mac OS & MacTeX & TeXShop \\
    \bottomrule
\end{tabular}
\end{table}
```

表 5.2: booktabs 宏包的效果

作業系統	發行版	編輯器
Windows	MikTeX	TeXnicCenter
Unix/Linux	TeX Live	Emacs
Mac OS	MacTeX	TeXShop

5.2 表格寬度

有時我們需要控制某欄位寬度，可以將其對齊方式參數從 l、c、r 改爲 p{寬度} 。

```
\begin{table}[htbp]
\caption{控制欄位寬度}
\centering
\begin{tabular}{p{100pt}p{100pt}p{100pt}}
    \toprule
    作業系統 & 發行版 & 編輯器 \\
    \midrule
    Windows & MikTeX & TeXnicCenter \\
    Unix/Linux & TeX Live & Emacs \\
    Mac OS & MacTeX & TeXShop \\
    \bottomrule
\end{tabular}
\end{table}
```

若想控制整個表格的寬度可以使用 tabularx 宏包，X 參數表示某欄可以折行。

表 5.3: 控制欄位寬度

作業系統	發行版	編輯器
Windows	MikTeX	TeXnicCenter
Unix/Linux	TeX Live	Emacs
Mac OS	MacTeX	TeXShop

```
\begin{table}[htbp]
\caption{控製表格寬度}
\centering
\begin{tabularx}{350pt}{lXlX}
    \toprule
    李白 & 平林漠漠煙如織，寒山一帶傷心碧。暝色入高樓，有人樓上愁。玉梯空佇立，宿鳥歸飛急。何處是歸程，長亭更短亭。&
    泰戈爾 & 夏天的飛鳥，飛到我的窗前唱歌，又飛去了。秋天的黃葉，它們沒有什麼可唱，只嘆息一聲，飛落在那裡。\\
    \bottomrule
\end{tabularx}
\end{table}
```

表 5.4: 控製表格寬度

李白	平林漠漠煙如織，寒山一帶傷心碧。暝色入高樓，有人樓上愁。玉階空佇立，宿鳥歸飛急。何處是歸程，長亭更短亭。	泰戈爾	夏天的飛鳥，飛到我的窗前唱歌，又飛去了。秋天的黃葉，它們沒有什麼可唱，只嘆息一聲，飛落在那裡。

5.3 跨行、跨列表格

有時某欄需要橫跨幾列，我們可以使用 `\multicolumn` 命令。它的前兩個參數指定橫跨列數和對齊方式。`booktabs` 宏包的 `\cmidrule` 命令用於橫跨幾列的

橫線。

```
\begin{table}[htbp]
\caption{跨欄表格}
\centering
\begin{tabular}{lll}
    \toprule
    & \multicolumn{2}{c}{常用工具} \\
    \cmidrule{2-3}
    作業系統 & 發行版 & 編輯器 \\
    \midrule
    Windows & MikTeX & TeXnicCenter \\
    Unix/Linux & TeX Live & Emacs \\
    Mac OS & MacTeX & TeXShop \\
    \bottomrule
\end{tabular}
\end{table}
```

表 5.5: 跨欄表格

	常用工具	
作業系統	發行版	編輯器
Windows	MikTeX	TeXnicCenter
Unix/Linux	TeX Live	Emacs
Mac OS	MacTeX	TeXShop

跨行表格需要使用 `multirow` 宏包，`\multirow` 命令的前兩個參數是豎跨的行數和寬度。

```
\usepackage{multirow}
...
\begin{table}[htbp]
\caption{跨行表格}
\centering
\begin{tabular}{lllc}
```

```
    \toprule
    作業系統 & 發行版 & 編輯器 & 用戶體驗\\
    \midrule
    Windows & MikTeX & TeXnicCenter &
    \multirow{3}{*}{\centering 爽} \\
    Unix/Linux & TeX Live & Emacs \\
    Mac OS & MacTeX & TeXShop \\
    \bottomrule
\end{tabular}
\end{table}
```

表 5.6: 跨行表格

作業系統	發行版	編輯器	用戶體驗
Windows	MikTeX	TeXnicCenter	
Unix/Linux	TeX Live	Emacs	爽
Mac OS	MacTeX	TeXShop	

5.4 彩色表格

彩色表格需要使用 `colortbl` 宏包[2]提供的一些命令：`\columncolor`、`\rowcolor`、`\cellcolor` 等。

```
\usepackage{colortbl}
...
\begin{table}[htbp]
\caption{彩色表格}
\centering
\begin{tabular}{lll}
    \toprule
    作業系統 & 發行版 & 編輯器 \\
    \midrule
    Windows & MikTeX & TeXnicCenter \\
```

```
    \rowcolor[gray]{.8} Unix/Linux & TeX Live & Emacs \\
    Mac OS & MacTeX & TeXShop \\
    \bottomrule
\end{tabular}
\end{table}
```

表 5.7: 彩色表格

作業系統	發行版	編輯器
Windows	MikTeX	TeXnicCenter
Unix/Linux	TeX Live	Emacs
Mac OS	MacTeX	TeXShop

5.5 長表格

有時表格太長要跨頁，可以使用 `longtable` 宏包[3]。`\endfirsthead`、`\endhead` 命令用來定義首頁表頭和通用表頭，`\endfoot`、`\endlastfoot` 命令用來定義通用表尾和末頁表尾。

```
\usepackage{longtable}
...
\begin{longtable}{ll}
\caption{長表格} \\
    \toprule
    作者 & 作品 \\
    \midrule
    \endfirsthead
    \midrule
    作者 & 作品 \\
    \midrule
    \endhead
    \midrule
    \multicolumn{2}{r}{接下頁\dots} \\
```

```
    \endfoot
    \bottomrule
    \endlastfoot
    白居易 & 漢皇重色思傾國，\\
    & 御宇多年求不得。\\
    & 楊家有女初長成，\\
    & 養在深閨人未識。\\
    & 天生麗質難自棄，\\
    & 一朝選在君王側。\\
    & 回眸一笑百媚生，\\
    & 六宮粉黛無顏色。\\
    & 春寒賜浴華清池，\\
    & 溫泉水滑洗凝脂。\\
    & 侍兒扶起嬌無力，\\
    & 始是新承恩澤時。\\
    & 雲鬢花顏金步搖，\\
    & 芙蓉帳暖度春宵。\\
    & 春宵苦短日高起，\\
    & 從此君王不早朝。\\
\end{longtable}
```

表 5.8: 長表格

作者	作品
白居易	漢皇重色思傾國， 御宇多年求不得。 楊家有女初長成， 養在深閨人未識。 天生麗質難自棄， 一朝選在君王側。 回眸一笑百媚生， 六宮粉黛無顏色。

接下頁...

作者	作品
	春寒賜浴華清池， 溫泉水滑洗凝脂。 侍兒扶起嬌無力， 始是新承恩澤時。 雲鬢花顏金步搖， 芙蓉帳暖度春宵。 春宵苦短日高起， 從此君王不早朝。

參考文獻

[1] Simon Fear. *Publication Quality Tables in LaTeX*, 2005. URL http://www.ctan.org/tex-archive/macros/latex/contrib/booktabs/.

[2] David Carlisle. *The colortbl Package*, 2001. URL http://tug.ctan.org/tex-archive/macros/latex/contrib/colortbl/.

[3] David Carlisle. *The longtable Package*, 2004. URL http://www.ctan.org/pkg/longtable.

看什麼看，沒見過空白頁？

第六章　雜項

6.1　超鏈接

hyperref 宏包[1]提供了一些超鏈接功能。它給文檔內部的交叉引用和參考文獻自動加上了超鏈接，還提供了幾個命令。

\hyperref 命令對已經定義的label進行簡單包裝，加上文字描述。

```
\usepackage{hyperref}
...
\label{sec:hyperlink}
...
例　如\ref{sec:hyperlink}是　編　號　形　式　的　鏈　接，
而\hyperref[sec:hyperlink]{這 個 鏈 接}是 文 字 形 式 的 鏈 接，都 指 向 本 節
開始。
```

例如6.1是編號形式的超鏈接，而這個鏈接則是文字形式，都指向本節開始。

\url 和 \href 命令可以用來定義外部鏈接，後者有文字描述。

```
\url{http://www.dralpha.com/}
\href{http://www.dralpha.com/}{包老師的主頁}
```

http://www.dralpha.com/
包老師的主頁

6.2 長文檔

當文檔很長時，我們可以把它分爲多個文件，然後在主控文檔的正文中引用它們。注意 `\include` 命令會新起一頁，如果不想要新頁可以改用 `\input` 命令。

```
%master.tex
\begin{document}
\include{chapter1.tex}
\include{chapter2.tex}
...
\end{document}
```

當文檔很長時，編譯一遍也會很花時間，我們可以用 `syntonly` 宏包。這樣編譯時就只檢查語法，而不生成結果文件。

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

6.3 參考文獻

在文檔中，我們經常要引用參考文獻（bibliography）。LaTeX 提供的 `thebibliography` 環境和 `\bibtem` 命令可以用來定義參考文獻條目及其列表顯示格式，`cite` 命令用來在正文中引用參考文獻條目。這種方法把內容和格式混在一起，用戶需要爲每個條目設置格式，很繁瑣且易出錯。

6.3.1 BibTeX

1985年， Oren Patashnik[1]和 Lamport 開發了 BibTeX[2]，其詳細使用方法請參閱 Nicolas Markey 的《Tame the BeaST: The B to X of BibTeX》[3]

BibTeX 把參考文獻的數據放在一個 `.bib` 文件中，顯示格式放在 `.bst` 文件中。普通用戶一般不需要改動 `.bst`，只須維護 `.bib` 數據庫。

[1]Wiki 上說他是 Knuth 的學生，我發現他不在 Knuth 的博士生列表上，而在姚期智的博士生列表上，也許他是 Knuth 的碩士生。

一個 .bib 文件可以包含多個參考文獻條目（entry），每個條目有類型、關鍵字，以及題目、作者、年份等字段。常用條目類型有 article、book、conference、manual、misc、techreport 等。每種類型都有一些自己的規定字段和可選字段，字段之間用逗號分開。數據庫中每個條目的關鍵字要保持唯一，因爲引用時要用到它們。

下例顯示了一個條目，它的類型是 manual，關鍵字是 Markey_2005。.bib 文件可以用普通文本編輯器來編輯，也可以用專門的文獻管理軟件來提高效率。包老師推薦 JabRef。

```
@MANUAL{Markey_2005,
  title = {Tame the BeaST: The B to X of BibTeX},
  author = {Nicolas Markey},
  year = {2005},
  url = {http://www.ctan.org/tex-archive/info/bibtex/
    tamethebeast/}
}
```

有了數據庫，我們可以像下面這樣引用一個條目。

```
請參閱\cite{Markey_2005}。
```

請參閱[3]。

前文中我們提到含有交叉引用的文檔需要編譯兩遍。含有參考文獻的文檔更麻煩，它需要依次執行 latex、bibtex、latex、latex 等四次編譯操作。

1. 第一遍 latex 只把條目的關鍵字寫到中間文件 .aux 中去。
2. bibtex 根據.aux、.bib、.bst 生成一個 .bbl 文件，即參考文獻列表。它的內容就是 thebibliography 環境和一些 \bibtem 命令。
3. 第二遍 latex 把交叉引用寫到 .aux 中去。
4. 第三遍 latex 則在正文中正確地顯示引用。

注意在長文檔中使用參考文獻時，應該用 latex 編譯主控文檔，而用 bibtex 編譯子文檔。

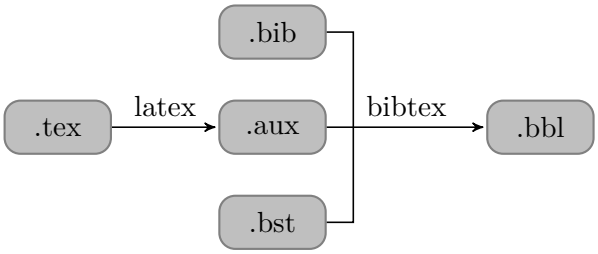


圖 6.1: BibTeX 的編譯

```
latex master(.tex)
bibtex chapter1(.tex)
latex master(.tex)
latex master(.tex)
```

6.3.2 natbib

參考文獻的引用通常有兩種樣式：作者-年份和數字。LaTeX 本身只支持數字樣式，而 `natbib` 宏包[4]則同時支持這兩種樣式。

使用 `natbib` 宏包時，我們首先要引用宏包；其次設置文獻列表樣式和引用樣式，每種列表樣式都有自己的預設引用樣式，所以後者可選；然後指定參考文獻數據庫。

```
\usepackage{natbib}
...
\begin{document}
\bibliographystyle{plainnat}
\setcitestyle{square,aysep={},yysep={;}}
\bibliography{mybib.bib}
...
\end{document}
```

`natbib` 提供了三種列表樣式：plainnat、abbrvnat、unsrtnat。前兩種都是作者-年份樣式，文獻列表按作者-年份排序，後者會使用一些縮寫（比如作者的 first name）；unsrtnat 是數字樣式，文獻列表按引用順序排序。

`\setcitestyle` 命令可以用來改變引用樣式的設置，其選項見 表 6.1。

表 6.1: 參考文獻引用樣式選項

引用模式	authoryear、numbers、super
括號	round、square、open=char,close=char
引用條目分隔符	分號、逗號、citesep=char
作者年份分隔符	aysep=char
共同作者年份分隔符	yysep=char
註解分隔符	notesep=text

注意在長文檔中，每個含參考文獻的子文檔都需要分別設置列表樣式，並指定數據庫。

natbib 提供了多種引用命令，其中最基本的是 \citet 和 \citep ，它們在不同引用模式下效果不同。一般不推薦使用 LaTeX 本身提供的 \cite，因爲它在作者-年份模式下和 \citet 一樣，在數字模式下和 \citep 一樣。

作者-年份模式下引用命令的效果如下。

```
參閱\cite{Daly_2007}\\
參閱\citet{Daly_2007}\\
參閱\citep{Daly_2007}
```

參閱Daly [2007]
參閱Daly [2007]
參閱[Daly, 2007]

數字模式下引用命令的效果如下。

```
參閱\cite{Daly_2007}\\
參閱\citet{Daly_2007}\\
參閱\citep{Daly_2007}
```

參閱[4]
參閱Daly [4]
參閱[4]

上標模式下引用命令的效果如下。

```
參閱\cite{Daly_2007}\\
參閱\citet{Daly_2007}\\
參閱\citep{Daly_2007}
```

參閱[4]
參閱Daly[4]
參閱[4]

另外還有一些引用命令，如 \citetext、\citenum、\citeauthor、\citeyear 等，此處不贅述。

6.4 索引

makeidx 宏包提供了索引功能。應用它時，我們首先需要在文檔序言部分引用宏包，並使用 makeindex 命令；其次在正文中需要索引的地方定義索引，注意索引關鍵字在全文中須保持唯一；最後在合適的地方（一般是文檔末尾）打印索引。

```
\usepackage{makeidx}
\makeindex
...
\begin{document}
\index{索引關鍵字}
...
\printindex
\end{document}
```

當編譯含索引的文檔時，用戶需要執行 latex、makeindex、latex 等三次編譯操作。

1. 第一遍 latex 把索引條目寫到一個 .idx 文件中去。
2. makeindex 把 .idx 排序後寫到一個 .ind 文件中去。
3. 第二遍 latex 在 \printindex 命令的地方引用 .ind 的內容，生成正確的DVI。

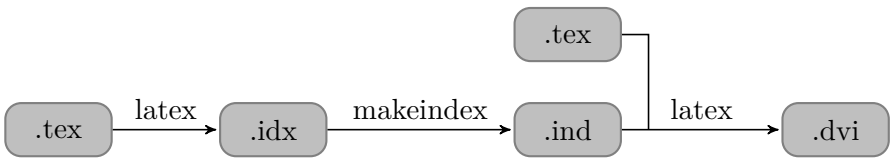


圖 6.2: 索引的編譯

6.5 頁面佈局

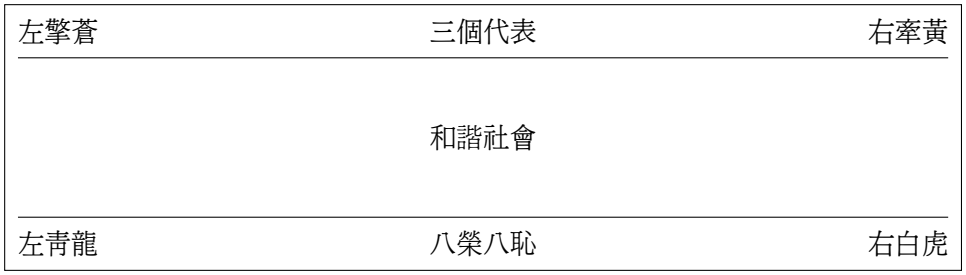
在 LaTeX 中用戶可以通過 \pagestyle 和 \pagenumbering 命令來設置頁眉（header）、頁脚（footer）的樣式和內容。頁面樣式有以下四種。

表 6.2: LaTeX 頁面樣式

empty	頁眉、頁脚空白
plain	頁眉空白，頁脚含居中頁碼
headings	頁脚空白，頁眉含章節名和頁碼
myheadings	頁脚空白，頁眉含頁碼和用戶自定義信息

fancyhdr[5]宏包提供了更靈活的控制。我們可以用以下代碼定製頁眉、頁脚的內容，以及頁眉下方、頁脚上方的橫線。

```
\usepackage{fancyhdr}
...
\pagestyle{fancy} %fancyhdr宏包新增的頁面風格
\lhead{左擎蒼}
\chead{三個代表}
\rhead{右牽黃}
\lfoot{左青龍}
\cfoot{八榮八恥}
\rfoot{右白虎}
\renewcommand{\headrulewidth}{0.4pt}
\renewcommand{\footrulewidth}{0.4pt}
```



用戶可以在頁眉、頁脚中使用一些 LaTeX 變量，比如分別代表頁碼和章節編號的 \thepage、\thechapter、\thesection；代表章節起始單詞 (Chapter、Section等) 的 \chaptername、\sectionname 等。

這些變量組合起來可以構成復合標記 \leftmark 和 \rightmark 。當文檔奇偶頁面佈局不同時，我們可以使用以下方法爲奇偶頁分別設置頁眉、頁脚。fancyhdr 宏包會自動把每章起始頁的樣式設爲 plain，若想去掉頁脚中間的頁碼，可以重定義 plain 樣式。

```
\pagestyle{fancy}
\fancyhf{}                      %清空頁眉頁腳
\fancyhead[LE,RO]{\thepage} %偶數頁左，奇數頁右
\fancyhead[RE]{\leftmark}   %偶數頁右
\fancyhead[LO]{\rightmark}  %奇數頁左
\fancypagestyle{plain}{     %重定義plain頁面樣式
    \fancyhf{}
    \renewcommand{\headrulewidth}{0pt}
}
```

Lamport當初設計 LaTeX 時把頁面佈局變量的定義方式搞得比較晦澀，用戶在重定義 \leftmark 和 \rightmark 時，不能直接用 \renewcommand 的方法，

而要用另外兩個命令。

```
\markboth{main-mark}{sub-mark}
\markright{sub-mark}
```

`\leftmark` 即 main-mark，是一種高層次標記，在 article 文檔類中它包含 section 的信息，在 report 和 book 則包含 chapter 的信息；`rightmark` 則是一種低層次標記，在 article 中包含 subsection 信息，在 report 和 book 中包含 section 信息。

比如在 `book` 文檔類中，章節標記是通過下面的方法定義的，其中的 `#1` 指的是章節的名字。

```
\renewcommand\chaptermark[1]{\markboth{\chaptername \thechapter.
    #1}{}}
\renewcommand\sectionmark[1]{\markright{\thesection. #1}}
```

參考文獻

[1] Sebastian Rahtz and Heiko Oberdiek. *Hypertext Marks in LaTeX: A Manual for hyperref*, 2006. URL http://www.tug.org/applications/hyperref/.

[2] Oren Patashnik. *BibTeXing*, 1988. URL http://www.ctan.org/tex-archive/biblio/bibtex/contrib/doc/.

[3] Nicolas Markey. *Tame the BeaST: The B to X of BibTeX*. CTAN, 2005. URL http://www.ctan.org/tex-archive/info/bibtex/tamethebeast/.

[4] Patrick W. Daly. *Natural Sciences Citations and References*, 2007. URL http://www.mps.mpg.de/software/latex/localtex/localltx.html.

[5] Piet van Oostrum. *Page Layout in LaTeX*, 2004. URL http://tug.ctan.org/tex-archive/macros/latex/contrib/fancyhdr/.

看什麼看，沒見過空白頁？

第七章　中文

7.1　字符集和編碼

衆所周知電腦內部採用二進制編碼，因爲它易於用電子電路實現。所有字符（包括字母、數字、符號、控制碼等）在電腦內部都是用二進製表示的，字符集（Character Set）的二進制編碼被稱爲字符編碼（Character Encoding），有時人們也會混用這兩個術語。

1963 年發佈的 American Standard Code for Information Interchange（ ASCII）是最早出現的字符編碼，它用 7 位（bit）表示了 $2^7 = 128$個字符，只能勉強覆蓋英文字符。

美國人發明了電腦，英語被優先考慮是很自然的事情。隨著電腦技術的傳播，人們呼籲把字符編碼擴充到 8 位也就是一個字節（byte），於是國際標準化組織（International Organization for Standardization，ISO）推出了 ISO 8859。$2^8 = 256$個字符顯然也不能滿足需要，所以 8859 被分爲十幾個部分，從 8859-1（西歐語言）、8859-2（中歐語言），直到 8859-16（東南歐語言），覆蓋了大部分使用拉丁字母的語言文字。

在 ISO 標準完全定型之前，IBM 就有一系列自己的字符編碼，他們稱之爲代碼頁（Code Page），其中著名的有 1981 年就被用於 IBM PC 的 437（擴展 ASCII）、850（西歐語言）、852（東歐語言）。IBM 代碼頁通常被用於控制台（Console）環境，也就是 MS-DOS 或 Unix Shell 那樣的命令行環境。

微軟將 IBM 代碼頁稱爲 OEM 代碼頁，自己定義的稱爲 ANSI 代碼頁，後者中著名的有1252（西歐語言）、1250（東歐語言）、936（GBK 簡體中文）、950（Big5 繁體中文）、932（SJIS 日文）、949（EUC-KR 韓文）等。

1981 年，中國大陸推出了第一個自己的字符集標準 GB2312，它是一個 94 * 94 的表，包括 7445 個字符（含 6763 個漢字）。GB2312 通常採用雙字節的EUC-CN編碼，所以後者也常常被稱爲 GB2312 編碼；其實 GB2312 還有另一種編碼方式 HZ，只是不常用。GB2312 不包含朱鎔基的「鎔」字，政府、新聞、出版、印刷等行業和部門在使用中感到十分不便，於是它在 1993 年被擴展爲 GBK，後者包括 21886 個字符（含 21003 個漢字），沒有形成正式標準。2000 年發佈的 GB18030 包含 70244 個字符（含 27533 個漢字），採用四字節編碼。GB18030 之前還出現過一個 GB13000，沒有形成氣候。

1990 年ISO 推出了通用字符集（Universal Character Set，UCS），即 ISO 10646，意圖一統江湖。它有兩種編碼：雙字節的 UCS-2 和四字節的 UCS-4。

ISO 之外還有個希望一統江湖的組織：統一碼聯盟（The Unicode Consortium），它於 1991 年推出了 Unicode 1.0。後來兩家組織意識到沒必要做重複工作，於是開始合併雙方的成果，攜手奔小康。從 Unicode 2.0 開始，Unicode 採用了與 ISO 10646-1 相同的編碼。

Unicode 主要有三種編碼：UTF-8、UTF-16、UTF-32。UTF-8 使用一至四個 8 位編碼。互聯網工程任務組（Internet Engineering Task Force， IETF）要求所有網絡協議都支持 UTF-8，互聯網電子郵件聯盟（Internet Mail Consortium，IMC）也建議所有電子郵件軟件都支持 UTF-8，所以它已成爲互聯網上的事實標準。UTF-16 用一或兩個 16 位編碼，基本上是 UCS-2 的超集，和 ASCII 不兼容。UTF-32 用一個 32 位編碼，它是 UCS-4 的一個子集。

7.2 中文解決方案

TEX 是基於單字節編碼的，因爲 Knuth 當初開發 TEX 時沒考慮那麼遠，也沒有現成的標準可以借鑑。

LATEX 對中文的支持主要有兩種方法：張林波[1]開發的 CCT 和 Werner Lemberg[2]開發的 CJK 宏包。早期 CCT 比較流行，新的 CCT 也可以和 CJK 配合使用，網上可以找到的最後更新是 2003 年的。CJK 是當前主流，它不僅支持

[1]中科院計算數學所某實驗室主任。

[2]1968 年生於奧地利，從維也納音樂學院獲得作曲、指揮、鋼琴、樂團管理、歌手教練等五個專業文憑，後自學中文和數學。曾任職於奧地利和德國多家劇院和樂團，現任德國科布倫茨某劇院指揮。

中日韓等東亞文字，還支持幾十種其他不同語言的多種編碼。

支持簡體中文的 LaTeX 發行版有吳凌雲[3]的 CTeX 和李樹鈞[4]的 ChinaTeX，繁體中文的有吳聰敏[5]、吳聰慧兄弟的 cwTeX 和蔡奇偉[6]的 PUTeX。後面兩個台灣的發行版包老師不熟悉，前面兩個大陸的發行版都包含 MikTeX、CCT、CJK、WinEdt 等。

7.3 CJK的使用

CJK 的作者寫的使用說明比較凌亂和晦澀，讀者可以參閱李果正的《我的CJK》[1]。

CJK 有兩個基本宏包：`CJK` 和 `CJKutf8`，後者面向 UTF-8 編碼。CJK 環境的一般使用方法如下：

```
\usepackage{CJK(utf8)}
...
\begin{document}
\begin{CJK}{<encoding>}{<family>}
...
\end{CJK}
\end{document}
```

簡體中文常用編碼是 GBK 和 UTF8 。family 是指宋體、楷體、隸書等，具體引用要看電腦上安裝了什麼字體。麻煩的是 GBK 和 UTF8 字體不通用，也就是說每種編碼需要自己的字體。CJK 自帶的 UTF8 簡體字體有 gbsn（宋體）和 gkai（楷體）。CTeX 提供的 GBK 字體有 song（宋體）、fs（仿宋）、kai（楷體）、hei（黑體）、li（隸書）、 you（幼圓）等。

注意使用 `CJKutf8` 宏包時，`CJK` 環境的編碼最好是 UTF8，否則可能遭遇不測。

下面是一個簡單的 GBK 示例和一個 UTF8 示例，注意用編輯器保存時要選則相應的編碼格式，比如前者用 ANSI，後者用 UTF-8。

[3]中科院應用數學所研究員。
[4]德國哈根函授大學研究員。
[5]台灣大學電機系學士，美國羅徹斯特大學經濟學博士，現任台灣大學經濟系教授。
[6]美國猶他大學計算機博士，現任台灣靜宜大學資訊工程系教授。

```
\documentclass{article}
\usepackage{CJK}
\begin{document}
\begin{CJK}{GBK}{song}
這是一個CJK例子，使用了GBK編碼和song字體。
\end{CJK}
\end{document}
```

```
\documentclass{article}
\usepackage{CJKutf8}
\begin{document}
\begin{CJK}{UTF8}{gbsn}
這是一個CJK例子，使用了UTF-8編碼和gbsn字體。
\end{CJK}
\end{document}
```

參考文獻

[1] 李果正. 我的*CJK*, 2004. URL http://edt1023.sayya.org/tex/mycjk/.

第八章　字體

關於字體有三個重要概念：glyph、typeface、font。glyph 通常被翻譯爲字形，也有翻譯爲字體的；typeface 是一個書法和印刷領域的概念，它通常被翻譯爲字體或書體；font 曾經和 typeface 混用，但現在一般用作電腦領域的概念，在中國大陸被翻譯爲字體，在台灣被翻譯爲字型。

上述翻譯的混亂令人十分無奈，包老師決定在本文中把它們分別翻譯爲字形、字樣、字體，以正視聽。

字形是一個字符的具體圖形表現形式，一個字符可以有多個字形，比如漢字中的「强／强」、「戶／户／戶」；字樣是一組相同風格樣式的字形的集合，比如中文字樣有宋、仿、楷、黑、隸、篆等；一種字樣可以對應電腦上的幾種字體。

8.1　字樣

拉丁字母的字樣主要有三大類：Serif（Roman）、Sans Serif 和 Monospace（Typewriter）。Serif 的筆畫邊緣部分有些裝飾，類似於中文的宋體、仿宋、楷體、魏書等。Sans Serif 的筆畫則是平滑的，類似於中文的黑體。Sans 這個詞來源於法語，就是「沒有」的意思。Monospace 則是等寬字樣。

每一類字樣都可以有加粗（bold）、斜體（italic）、傾斜（oblique）等修飾效果。Italic 通常對原字樣進行過重新設計，它修飾精細，多用於 Serif；Oblique 也稱作 slanted，基本上是把正體傾斜，多用於 Sans Serif。通常 oblique 看起來比 italic 要寬一些。

表 8.1列出了幾種常見的字樣。

表 8.1: 常見字樣

作業系統	Serif	Sans Serif	Monospace
Mac OS	Times	Helvetica	Courier
Windows	Times New Roman	Arial	Courier New

8.2 字體格式

8.2.1 點陣字體和向量字體

電腦上用的字體（font）按數據格式可以分爲三大類：點陣字體（bitmap）、輪廓（outline）字體和筆畫（stroke-based）字體。

點陣字體通過點陣來描述字形。早期的電腦受到容量和繪圖速度的限制，多採用點陣字體。點陣字體後來漸漸被輪廓字體所取代，但是很多小字號字體仍然使用它，因爲這種情況下輪廓字體縮放太多會導致筆畫不清晰。

輪廓字體又稱作向量字體，它通過一組直線段和曲線來描述字形。輪廓字體易於通過數學函數進行縮放等變換，形成平滑的輪廓。輪廓字體的主要缺陷在於它所採用的貝茲曲線（Bézier curves）在光柵（raster）設備（比如顯示器和打印機）上不能精確渲染，因而需要額外的補償處理比如字體微調（font hinting）。但是隨著電腦硬件的發展，人們一般不在意它比點陣字體多出的處理時間。

筆畫字體其實也是輪廓字體，不過它描述的不是完整的字形，而是筆畫。它多用於東亞文字。

8.2.2 常見字體

常見的輪廓字體技術有：Type 1 和 Type 3、TrueType、OpenType、METAFONT 等。

Adobe 的 Type 1 和 Type 3 基於 PS，它們採用三次貝茲曲線。Type 1 支持微調，它使用一個簡化的 PS 子集；Type 3 不支持微調，但它可以使用全部 PS 功能，因此既可以包含輪廓字體也可以包含點陣字體信息。

1991 年，Apple 發佈了 TrueType，它採用二次貝茲曲線。二次曲線處理起來比三次曲線快，但是需要更多的點來描述。所以從 TrueType 到 Type 1的轉

換是無損的，反之是有損的。1994 年，Apple 著手研究 TrueType 的下一代技術：TrueType GX，它後來演變爲Apple Advanced Typography（AAT）。

1996 年，微軟和 Adobe 聯合發佈了 OpenType。它比起 AAT 的優勢有：跨平台、開放和易於開發、支持更多的語言比如阿拉伯語。

早在 1984 年 Knuth 就發佈了 METAFONT，它與 TrueType 和 OpenType的區別是，不直接描述字形輪廓，而描述生成輪廓的筆的軌跡。筆的形狀可以是橢圓形或多邊形，尺寸縮放自如，字形邊緣也柔和一些。兩種字體可以用同一個 METAFONT 文件，當然還有不同的參數。METAFONT 技術如此先進，卻沒有流行開來。對此 Knuth 解釋道，要求一位設計字體的藝術家掌握 60 個參數太變態了，那是用來折磨數學家的。

Type 1 和 Type 3 把字體信息存儲在兩種文件裡：metrics 和 glyph 文件。metrics 文件有 AFM（Adobe font metrics）和 PFM（printer font metrics），glyph 文件有 PFA（printer font ASCII）和 PFB（printer font binary）。LaTeX 使用的 metrics 格式是 TFM（TeX Font Metrics）。

TrueType 的文件後綴是 .ttf，OpenType 的是 .ttf 和 .otf。METAFONT 雖然用向量圖形來定義字形，實際輸出的卻是一種點陣格式：PK （packed raster）。

上述字體按技術的先進性，從高到低的排序爲：OpenType、 TrueType、Type 1、Type 3、PK，我們應優先選用 OpenType 和 TrueType。

8.2.3 合縱連橫

Adobe 收取的 Type 1 專利許可費一度十分昂貴，窮人們只好用免費的 Type 3。爲了打破這種壟斷，Apple 開發了 TrueType。1991 年 TrueType 發佈之後，Adobe隨即公開了 Type 1 的規範， Type 1 字體從貴族墮落爲平民，因而流行開來。

1980 年代中後期，Adobe 的大部分盈利來自於 PS 解釋器的許可費。面對這種壟斷局面，微軟和 Apple 聯合了起來。微軟把買來的 PS 解釋器 TrueImage 授權給 Apple，Apple 則把 TrueType 授權給微軟。

微軟得隴望蜀，又企圖獲得 AAT 的許可證，未遂。爲了打破 Apple 的壟斷，微軟聯合 Adobe 在1996年發佈了 OpenType。Adobe 在2002 年末將其字體庫全面轉向 OpenType。

上面這幾齣精彩好戲充分展示了商場上的勾心鬥角、爾虞我詐，沒有永恆的夥伴，只有永恆的利益。但它同時也告訴我們，市場競爭中受益的還是廣大的消費者。

8.3 字體應用

PS 支持 Type 1 和 Type 3，而 PDF 除了這兩種還支持 TrueType 和 OpenType。`latex`、DVI瀏覽器、各種 driver 分別採用不同的字體技術。

8.3.1 DVI

`latex` 編譯 LaTeX 源文件生成 DVI 時只需要`.tfm` 文件，因爲 DVI 並不包含字形信息，而只包含對字體的引用。DVI 瀏覽器顯示 DVI 時一般使用 PK，它在系統中查找相應的 `.pk` 文件，若找不到就調用 METAFONT 在後台自動生成。

8.3.2 dvips

預設情況下，`dvips` 也會查找 `.pk` ，或調用 METAFONT 自動生成；然後把 PK 轉換成包含點陣字體的 Type 3，它的參數 `-D` 可以用來控制該點陣字體的分辨率。用 `ps2pdf` 處理含 Type 3 的 PS 時，輸出的自然是含 Type 3 的 PDF。

GSview 在低分辨率下可以很好地渲染 Type 3，Adobe Reader 或 Acrobat 卻不能，因爲它們使用的 Adobe Type Manager 不支持包含完整 PS 的 Type 3。含 Type 3 的 PDF 看起來會有些模糊，所以應儘量避免使用。

`dvips` 的另一個參數 `-Ppdf` 把 Type 1 嵌入生成的 PS，這樣再 `ps2pdf` 就能生成含 Type 1 的 PDF。

`dvips` 不支持眞正的（native）TrueType，用戶只能把 TrueType 先轉成 PK 或 Type 1，這樣繞了個彎效果總會打些折扣。

`dvips` 的字體詳細使用方法可查閱其手册[1]第 6 章，此處不贅述。

8.3.3 dvipdfm(x)

`dvipdfm` 支持 PK 和 Type 1，它可以用一個 `t1fonts.map` 文件建立 PK 文件和 Type 1 文件之間的映射，這樣生成的 PDF 用的就是 Type 1。`dvipdfm` 也

不支持眞正的 TrueType。

dvipdfmx 通過正確的設置可以使用眞正的 TrueType，它對中日韓等東亞文字的支持也較好，所以它對我們來說是 Driver 的首選。

8.4 TrueType 字體安裝配置

CJK 自帶的 UTF-8 編碼字體 gbsn 和 gkai只包含 GB2312 字符集，而 CTeX 只提供 GBK 編碼字體，因此中文用戶通常需要自己安裝配置 UTF-8 編碼的 TrueType 字體。

在使用 TrueType 之前，用戶通常需要作以下準備工作：

1. 用轉換程序 ttf2tfm 生成 TFM。
2. 配置字體定義文件 .fd。
3. 配置 ttf2pk ，因爲 DVI 瀏覽器和 dvips 都會自動調用 ttf2pk 來生成 PK。
4. 配置 dvipdfmx。

8.4.1 目錄和文件

通常每個發行包都會參照 TDS 建立自己的目錄系統，把各種文件發在固定的位置。比如 MiKTeX 頂層目錄如下，在本節後面的示例中我們將使用這些目錄的縮寫。

```
Install: D:\edit\MiKTeX 2.7
UserData: C:\Documents and Settings\Alpha\Local Settings\
    Application Data\MiKTeX\2.7
UserConfig: C:\Documents and Settings\Alpha\
    Application Data\MiKTeX\2.7
```

目錄多了有個缺點，文件不知道放在哪裡好。MiKTeX 中有的配置文件居然在四個目錄下各有一份，實在是令人髮指。幸好我們可以用下面的命令檢查配置文件的具體名字和路徑。

```
initexmf --edit-config-file=ttf2pk
```

8.4.2 ttf2tfm

比如我們想把 SimSun18030.ttc (18030 字符集的新宋體) 轉換爲 UTF8 編碼的字體文件，我們需要執行以下步驟。

1. 把需要的 .ttf 文件複製到 UserData/fonts/truetype/chinese/。
2. 用下面的命令生成.tfm 和 .enc 文件。
3. 把 *.tfm 複製到 UserData/fonts/tfm/chinese/utf8song/。
4. 把 *.enc 複製到 UserData/fonts/enc/chinese/utf8song/。

```
ttf2tfm SimSun18030.ttc -q -w utf8song@Unicode@
```

8.4.3 字體定義文件

字體定義文件將字體引用名和實際的字體文件聯繫起來，比如我們在 CJK 環境中引用 usong 時，系統將會找到並使用 utf8song*.tfm。

```
%UserData\tex\latex\CJK\UTF8\C70usong.fd
\ProvidesFile{c70usong.fd}
%character set: GB18030
%font encoding: Unicode
\DeclareFontFamily{C70}{usong}{\hyphenchar \font\m@ne}
\DeclareFontShape{C70}{usong}{m}{n}{<-> CJK * utf8song}{}
\DeclareFontShape{C70}{usong}{m}{it}{<-> CJK * utf8song}{}
\DeclareFontShape{C70}{usong}{bx}{n}{<-> CJKb * utf8song}{
    \CJKbold}
\endinput
```

8.4.4 配置 ttf2pk

MiKTeX 中 ttf2pk 的配置文件是 ttf2pk.ini ，在其它的發行包中可能是 ttf2pk.cfg。

ttf2pk.ini 中有一個 .map 文件列表，後者定義了 TrueType 應該按編碼轉爲 PK 等信息。

比如下面這個文件列表會讓 ttf2pk 讀取 foo.map 和 bar.map。

```
map foo.map
map bar.map
```

如果係統找不到 ttf2pk.ini，它會預設使用 ttfonts.map。

```
%UserData\ttf2tfm\base\ttfonts.map
utf8song@Unicode@ SimSun18030.ttc
```

8.4.5 配置 dvipdfmx

配置 dvipdfmx 是爲了讓 PDF 正確地嵌入 TrueType，否則生成的文件中的內容不能複製、黏貼。

```
%UserConfig\dvipdfm\config\dvipdfmx.cfg
f cid-x.map
```

```
%UserData\dvipdfm\config\cid-x.map
utf8song@Unicode@ unicode SimSun18030.ttc
```

參考文獻

[1] Tomas Rokicki. *Dvips: A DVI-to-PostScript Translator*, 2005. URL http://tug.org/texinfohtml/dvips.html.

看什麼看，沒見過空白頁？

跋

首先向一路披荆斬棘看到這裡的讀者表示祝賀，至少在精神上你已經成爲一名合格的 LaTeXer。從此你生是 LaTeX 的人，死是 LaTeX 的鬼。Once Black, never back。沒有堅持到這裡的同學自然已經重新投向「邪惡」的 MS Word，畢竟那裡點個按鈕就可以插入圖形，點個下拉框就可以選擇字體。

當然 LaTeXer也有簡單的出路，就是只使用預設設置，儘量少用插圖；不必理會點陣、向量，也不必理會 Type 1、Type 3、TrueType、OpenType。因爲內容高於形式，你把文章的版面、字體搞得再漂亮，它也不會因此成爲《紅樓夢》；而《紅樓夢》即使是手抄本，也依然是不朽的名著。

包老師曾經以爲 LaTeX 和 Word 的關係就好像是《笑傲江湖》中華山的氣宗和劍宗，頭十年劍宗進步快，中間十年打個平手，再往後氣宗就遙遙領先。至於令狐沖的無招勝有招，風清揚的神龍見首不見尾又是另一重境界，普通人恐怕只能望其頸背。

費盡九牛二虎之力熬到本文殺青的時候，才發現從前的想法很傻很天眞。讓我們揮一揮衣袖，不帶走一片雲，臥薪嘗膽忍辱負重，耐心等待 XeTeX 和 LuaTeX。