

# L<sup>A</sup>T<sub>E</sub>X Notes v 1.20

Alpha Huang<sup>1</sup>

2008年7月26日

<sup>1</sup><http://www.dralpha.com/>

看什么看，没见过空白页？

# 序

满纸荒唐言，一把辛酸泪！都云作者痴，谁解其中味？<sup>1</sup>

— 曹雪芹

最早听说 L<sup>A</sup>T<sub>E</sub>X 大约是 2002 年，一位同事演示了用它排版的一篇文章和几幅图。包老师<sup>2</sup>不以为然，因为那些东西用 Microsoft Word 和 Visio 也可以做到，而且可以做得更快。再次听说它是王垠同学在闹退学，传说他玩 Linux 和 L<sup>A</sup>T<sub>E</sub>X 而走火入魔。

大约是 2005 年底，看了一下 lshort，用 L<sup>A</sup>T<sub>E</sub>X 记了些数学笔记，开始有点感觉。包老师生性愚钝，所以喜欢相对简单的东西。HTML、Java 都用手写，FrontPage、Dreamweaver、JBuilder 之类笨重的家伙看两眼就扔了，所以喜欢上 L<sup>A</sup>T<sub>E</sub>X 只是时间问题。

次年老妻要写博士论文，拿出 Word 底稿让我排版。大家都知道 Word 太简单了，谁都能用，但是不是谁都能用好。人称电脑杀手的老妻制作的 Word 文档自然使出了各种奇门遁甲，加上她实验室、学校和家里电脑里的三个 EndNote 版本互不兼容，实在难以驯服。我只好重起炉灶，拿她的博士论文当小白鼠，试验一下 L<sup>A</sup>T<sub>E</sub>X 的威力。

就这样接触了两三年，总算略窥门径，感觉 L<sup>A</sup>T<sub>E</sub>X 实在是博大精深，浩如烟海。而人到中年大脑储存空间和处理能力都有点捉襟见肘，故时常作些笔记。一来对常用资料和问题进行汇编索引，便于查询；二来也记录一些心得。

---

<sup>1</sup>当年作博士论文时虽不曾增删五次披阅十载，也被折磨得欲仙欲死，故与室友戏言将此五绝加入序言。多年以后的今天终于实现了此夙愿。

<sup>2</sup>吾有多重人格，比如本色的是阿黄，下围棋的是隐忍灰衣人，道貌岸然的是包老师。

日前老妻吵着要学 L<sup>A</sup>T<sub>E</sub>X，便想这份笔记对初学者或有些许借鉴意义，于是系统地整理了一番，添油加醋包装上市。

原本打算分九章，以纪念《九章算术》，实际上第八章完成时已如强弩之末，最后一章还须另择黄道吉日。

本文第一章谈谈历史背景；第二章介绍入门基础；第三至五章讲解数学、插图、表格等对象的用法；第六章是一些特殊功能；第七、八章讨论中文和字体的处理；第九章附加定制内容。

从难易程度上看前两章较简单，插图、字体两章较难。一般认为 L<sup>A</sup>T<sub>E</sub>X 相对于微软的傻瓜型软件比较难学，所以这里采取循序渐进，温水煮青蛙的方法。

初则示弱，麻痹读者；再则巧言令色，请君入瓮；三则舌绽莲花，诱敌深入；彼入得罄中则摧动机关，关门打狗；继而严刑拷打，痛加折磨；待其意乱情迷彷徨无计之时，给予当头棒喝醍醐灌顶，虽戛然而止亦余音绕梁。

鄙人才疏学浅功力不逮，面对汗牛充栋罄竹难书<sup>3</sup>的资料，未免考虑不周挂一漏万，或有误导，敬请海涵。若有高手高手高高手略拨闲暇指点一二，在下感激不尽<sup>4</sup>。

借此感谢一下老妻，如果不是伊天天看韩剧，包老师也不会有时间灌水 and 整理这份笔记。

---

<sup>3</sup>此处用法循阿扁古例。

<sup>4</sup>[huang.xingang@gmail.com](mailto:huang.xingang@gmail.com)

# 目 录

序	iii
<b>1 简介</b>	<b>1</b>
1.1 历史回顾 . . . . .	1
1.2 优点和缺点 . . . . .	3
1.3 软件准备 . . . . .	4
1.4 学习方法 . . . . .	5
<b>2 入门</b>	<b>7</b>
2.1 Hello, World! . . . . .	7
2.2 格式及其转换 . . . . .	8
2.2.1 页面描述语言 . . . . .	8
2.2.2 格式转换 . . . . .	10
2.3 L <sup>A</sup> T <sub>E</sub> X 语句 . . . . .	11
2.4 文档结构 . . . . .	11
2.4.1 文档类、序言、正文 . . . . .	11
2.4.2 标题、摘要、章节 . . . . .	12
2.4.3 目录 . . . . .	13
2.5 文字排版 . . . . .	14
2.5.1 字符输入 . . . . .	14
2.5.2 换行、换页、断字 . . . . .	14
2.5.3 字样、字号 . . . . .	15

2.6	常用命令环境	16
2.6.1	列表	16
2.6.2	对齐	17
2.6.3	摘录	17
2.6.4	原文照排	18
2.6.5	交叉引用	19
2.6.6	脚注	19
2.7	长度单位	19
2.8	盒子	20
2.8.1	mbox 和 fbox	20
2.8.2	makebox 和 framebox	20
2.8.3	parbox 和 minipage	20
<b>3</b>	<b>数学</b>	<b>23</b>
3.1	数学模式	23
3.2	基本原素	24
3.2.1	字母	24
3.2.2	指数、下标、根号	24
3.2.3	分数	25
3.2.4	运算符	25
3.2.5	分隔符	26
3.2.6	箭头	26
3.2.7	标注	27
3.2.8	省略号	28
3.2.9	空白间距	28
3.3	矩阵和行列式	28
3.4	多行公式	29
3.4.1	长公式	29
3.4.2	公式组	29
3.5	定理和证明	30
3.6	数学字体	31

<b>4</b>	<b>插图</b>	<b>33</b>
4.1	图形格式	33
4.1.1	EPS	34
4.1.2	Driver 们的口味	34
4.1.3	图形格式转换	35
4.2	插入图形	36
4.2.1	插入命令	36
4.2.2	缩放、旋转	37
4.2.3	figure环境	38
4.2.4	插入多幅图形	39
4.3	图形绘制工具比较	42
4.4	METAPOST	42
4.4.1	准备工作	43
4.4.2	基本图形对象	44
4.4.3	点和线宽	45
4.4.4	图形控制	46
4.4.5	编程功能	49
4.5	PSTricks	51
4.5.1	准备工作	51
4.5.2	基本图形对象	52
4.5.3	图形控制	55
4.5.4	对象布局	57
4.6	PGF	58
4.6.1	准备工作	58
4.6.2	基本图形对象	59
4.6.3	图形控制	60
4.6.4	样式	61
4.6.5	流程图	62
<b>5</b>	<b>表格</b>	<b>65</b>
5.1	简单表格	65
5.2	表格宽度	67

5.3	跨行、跨列表格 . . . . .	68
5.4	彩色表格 . . . . .	70
5.5	长表格 . . . . .	71
<b>6</b>	<b>杂项</b>	<b>75</b>
6.1	超链接 . . . . .	75
6.2	长文档 . . . . .	76
6.3	参考文献 . . . . .	76
6.3.1	BibTeX . . . . .	76
6.3.2	natbib . . . . .	78
6.4	索引 . . . . .	80
6.5	页面布局 . . . . .	81
<b>7</b>	<b>中文</b>	<b>85</b>
7.1	字符集和编码 . . . . .	85
7.2	中文解决方案 . . . . .	86
7.3	CJK的使用 . . . . .	87
<b>8</b>	<b>字体</b>	<b>89</b>
8.1	字样 . . . . .	89
8.2	字体格式 . . . . .	90
8.2.1	点阵字体和矢量字体 . . . . .	90
8.2.2	常见字体 . . . . .	90
8.2.3	合纵连横 . . . . .	91
8.3	字体应用 . . . . .	92
8.3.1	DVI . . . . .	92
8.3.2	dvips . . . . .	92
8.3.3	dvipdfm(x) . . . . .	93
8.4	TrueType 字体安装配置 . . . . .	93
8.4.1	目录和文件 . . . . .	93
8.4.2	ttf2tfm . . . . .	94
8.4.3	字体定义文件 . . . . .	94
8.4.4	配置 ttf2pk . . . . .	95



8.4.5 配置 dvipdfmx . . . . . 95

跋 97

看什么看，没见过空白页？

# 第一章 简介

滚滚长江东逝水，浪花淘尽英雄。是非成败转头空。青山依旧在，几度夕阳红。

白发渔樵江渚上，惯看秋月春风。一壶浊酒喜相逢。古今多少事，都付笑谈中。

— 杨慎《临江仙》

## 1.1 历史回顾

L<sup>A</sup>T<sub>E</sub>X 是一种面向数学和其它科技文档的电子排版系统。一般人们提到的 L<sup>A</sup>T<sub>E</sub>X 是一个总称，它包括 T<sub>E</sub>X、L<sup>A</sup>T<sub>E</sub>X、 $\mathcal{A}\mathcal{M}\mathcal{S}$ -L<sup>A</sup>T<sub>E</sub>X 等<sup>1</sup>。

T<sub>E</sub>X 的开发始于 1977 年 5 月，Donald E. Knuth<sup>2</sup>开发它的初衷是用于《The Art of Computer Programming》的排版。1962 年 Knuth 开始写一本关于编译器设计的书，原计划是 12 章的单行本。不久 Knuth 觉得此书涉及的领域应该扩大，于是越写越多，如滔滔江水连绵不绝，又如黄河泛滥一发不可收拾。1965 年完成的初稿居然有 3000 页，全是手写的！据出版商估计，这些手稿印刷出来需要 2000 页，出书的计划只好改为七卷，每卷一或两章。1976 年 Knuth 改写第二卷的第二版时，很郁闷地发现第一卷的铅版不见了，而当时电子排版刚刚兴起，质量还差强人意。于是 Knuth 仰天长啸：“我要扼住命运的咽喉”，决定自己开发一个全新的系统，这就是 T<sub>E</sub>X。

---

<sup>1</sup>一般认为 T<sub>E</sub>X 是一种引擎，L<sup>A</sup>T<sub>E</sub>X 是一种格式，而  $\mathcal{A}\mathcal{M}\mathcal{S}$ -L<sup>A</sup>T<sub>E</sub>X 等是宏集。此处目的是简介，故不展开讨论。

<sup>2</sup>斯坦福大学计算机系教授，已退休。

1978 年  $\text{T}_{\text{E}}\text{X}$  第一版发布后好评如潮，Knuth 趁热打铁在 1982 年发布了第二版。人们现在使用的  $\text{T}_{\text{E}}\text{X}$  基本就是第二版，中间只有一些小的改进。1990 年  $\text{T}_{\text{E}}\text{X}$  v3.0 发布后，Knuth 宣布除了修正 bug 外停止  $\text{T}_{\text{E}}\text{X}$  的开发，因为他要集中精力完成那本巨著的后几卷<sup>3</sup>。此后每发布一个修正版，版本号就增加一位小数，使得它趋近于  $\pi$ （目前是 3.141592）。Knuth 希望将来他离世时， $\text{T}_{\text{E}}\text{X}$  的版本号永远固定下来，从此人们不再改动他的代码。他开发的另一个软件 METAFONT 也作类似处理，它的版本号趋近于  $e$ ，目前是 2.71828。

$\text{T}_{\text{E}}\text{X}$  是一种语言也是一个宏处理器，这使得它很好很强大，但是它同时又很繁琐，让人难以接近。因此 Knuth 提供了一个对  $\text{T}_{\text{E}}\text{X}$  进行了封装的宏集 Plain  $\text{T}_{\text{E}}\text{X}$ ，里面有一些高级命令，有了它最终用户就无须直接面对枯燥无味的  $\text{T}_{\text{E}}\text{X}$ 。

然而 Plain  $\text{T}_{\text{E}}\text{X}$  还是不够高级，所以 Leslie Lamport<sup>4</sup>在 80 年代初期开发了另一个基于  $\text{T}_{\text{E}}\text{X}$  的宏集  $\text{L}^{\text{A}}\text{T}_{\text{E}}\text{X}$ 。1992 年  $\text{L}^{\text{A}}\text{T}_{\text{E}}\text{X}$  v2.09 发布后，Lamport 退居二线，之后的开发活动由 Frank Mittelbach 领导的 The LaTeX Team 接管。此小组发布的最后版本是 1994 年的  $\text{L}^{\text{A}}\text{T}_{\text{E}}\text{X} 2_{\epsilon}$ ，他们同时还在进行  $\text{L}^{\text{A}}\text{T}_{\text{E}}\text{X} 3$  的开发，只是正式版看起来遥遥无期。

起初，美国数学学会（American Mathematical Society, AMS）看着  $\text{T}_{\text{E}}\text{X}$  是好的，就派 Michael Spivak 写了  $\mathcal{A}\mathcal{M}\mathcal{S}\text{-T}_{\text{E}}\text{X}$ ，这项基于 Plain  $\text{T}_{\text{E}}\text{X}$  的开发活动进行了两年（1983–1985）。后来与时俱进的 AMS 又看着  $\text{L}^{\text{A}}\text{T}_{\text{E}}\text{X}$  是好的，就想转移阵地，但是他们的字体遇到了麻烦。恰好 Mittelbach 和 Rainer Schöpf（后者也是 LaTeX Team 的成员）刚刚发布了 New Font Selection Scheme for  $\text{L}^{\text{A}}\text{T}_{\text{E}}\text{X}$ （NFSS），AMS 看着还不错，就拜托他们把 AMSFonts 加入  $\text{L}^{\text{A}}\text{T}_{\text{E}}\text{X}$ ，继而在 1989 年请他们开发  $\mathcal{A}\mathcal{M}\mathcal{S}\text{-L}^{\text{A}}\text{T}_{\text{E}}\text{X}$ 。 $\mathcal{A}\mathcal{M}\mathcal{S}\text{-L}^{\text{A}}\text{T}_{\text{E}}\text{X}$  发布于 1990 年，之后它被整合为  $\mathcal{A}\mathcal{M}\mathcal{S}$  宏包，像其它宏包一样可以直接运行于  $\text{L}^{\text{A}}\text{T}_{\text{E}}\text{X}$ 。

---

<sup>3</sup>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》正在写作中，预计 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。

<sup>4</sup>现供职于微软研究院。

## 1.2 优点和缺点

当前的文字处理系统大致可以分为两种：标记语言（Markup Language）式的，比如 $\text{\LaTeX}$ ；所见即所得（WYSIWYG）式的，比如 MS Word<sup>5</sup>。

一般而言， $\text{\LaTeX}$  相对于所见即所得系统有如下优点：

- 高质量 它制作的版面看起来更专业，数学公式尤其赏心悦目。
- 结构化 它的文档结构清晰。
- 批处理 它的源文件是文本文件，便于批处理，虽然解释（parse）源文件可能很费劲。
- 跨平台 它几乎可以运行于所有电脑硬件和操作系统平台。
- 免费 多数  $\text{\LaTeX}$  软件都是免费的，虽然也有一些商业软件。

相应地， $\text{\LaTeX}$  的工作流程、设计原则，资源的缺乏，以及开发人员的历史局限性等种种原因也导致了一些缺陷：

- 制作过程繁琐，有时需要反复编译，不能直接或实时看到结果。
- 宏包鱼龙混杂，水准参差不齐，风格不够统一。
- 排版风格比较统一，但因而缺乏灵活性。
- 用户支持不够好，文档不完善。
- 对国际语言和字体的支持很差。

抛开 MS Word 不谈，即使跟同为标记语言的 HTML/Web 系统相比， $\text{\LaTeX}$  也有一些不足之处。比如 Web 浏览器对 HTML 内容的渲染（render）比 DVI 浏览器对  $\text{\LaTeX}$  内容的渲染要快上许多，基本上可以算是实时。虽然 HTML 内容可能没有 LaTeX 那么复杂，但是 DVI 毕竟是被  $\text{\LaTeX}$  编译过的格式。

---

<sup>5</sup>其实 Word 也有自己的标记语言域代码（field code），只是一般用户不了解。

还有一点令人困惑的是，有一部分  $\text{\LaTeX}$  阵营的人士习惯于称对方为“邪恶的”或“出卖灵魂的”，如果昂贵的微软系统应当为人诟病，那么更贵的苹果系统为何却被人追捧？

2000 年有记者在采访 Lamport 时问：“为什么当前没有高质量的所见即所得排版系统？”他回答道：“门槛太高了，一个所见即所得系统要做到  $\text{\LaTeX}$  当前的水平，工作量之大不是单枪匹马所能完成<sup>6</sup>。微软那样的大公司可以做，但是市场太小了。我偶尔也会想加入“Dark Side”，让微软给我一组人马来开发一个这样的系统。”（包老师注：他果然于次年加入微软。）

窃以为这两大阵营其实是萝卜青菜的关系，与其抱残守缺、互相攻讦，不如各取所需；甚至可以捐弃前嫌、取长补短，共建和谐社会。

### 1.3 软件准备

$\text{\LaTeX}$  是一个软件系统，同时也是一套标准。遵照这些标准，实现了（implement）所要求功能的软件集合被称为发行版（distribution）。与此类似的例子有 Java 和 Linux，比如 SUN、IBM、BEA 等公司都有自己的 Java 虚拟机（JVM），它们都被称作 Java 的实现；而 Linux 有 Red Hat/Fedora、Ubuntu、SuSE 等众多的发行版。

表 1.1:  $\text{\LaTeX}$  发行版与编辑器

操作系统	发行版	编辑器
Windows	<a href="#">MikTeX</a>	<a href="#">TeXnicCenter</a> 、 <a href="#">WinEdt</a>
Unix/Linux	<a href="#">TeX Live</a>	<a href="#">Emacs</a> 、 <a href="#">vim</a> 、 <a href="#">Kile</a>
Mac OS	<a href="#">MacTeX</a>	<a href="#">TeXShop</a>

$\text{\LaTeX}$  发行版只提供了一个  $\text{\LaTeX}$  后台处理机制，用户还需要一个前台编辑器来编辑它的源文件。常用的  $\text{\LaTeX}$  发行版和编辑器见表 1.1。在使用  $\text{\LaTeX}$  的过程中可能还需要其它一些软件，将在后面相关章节中分别介绍。

<sup>6</sup> $\text{\TeX}/\text{\LaTeX}$  也不单单是那几个大腕儿完成的，他们背后还有众多默默无闻的小人物，比如当年 Knuth 手下的大批学生。此所谓一将功成万骨枯。

## 1.4 学习方法

在科学上没有平坦的大道，只有那些不畏劳苦沿着陡峭山路攀登的人，才有希望达到光辉的顶点。

— 卡尔·马克思

无他，唯手熟尔。

— 卖油翁

用心。

— 斯蒂芬·周

限于篇幅和水平，本文只能提供一个概览外加一些八卦。比较严谨的入门资料有 Tobias Oetiker 的《A (Not So) Short Introduction to L<sup>A</sup>T<sub>E</sub>X 2<sub>ε</sub>》<sup>[1]</sup>（简称lshort）；若想对 L<sup>A</sup>T<sub>E</sub>X 有更深入全面的了解，可以拜读 Mittelbach 的《The L<sup>A</sup>T<sub>E</sub>X Companion》<sup>[2]</sup>。

中文资料可参考李果正的《大家来学 L<sup>A</sup>T<sub>E</sub>X》<sup>[3]</sup>，lshort 有吴凌云等人翻译的中文版本<sup>7</sup>。

[Comprehensive TeX Archive Network](#)（CTAN）和 [TeX Users Group](#)（TUG）提供了权威、丰富的资源。

英国TUG 和 C<sup>T</sup>E<sub>X</sub> 分别提供了常见问题集（FAQ）<sup>[4;5]</sup>，一般问题多会在这里找到答案。

中文 T<sub>E</sub>X 论坛有[水木清华 BBS TeX 版](#)、[C<sup>T</sup>E<sub>X</sub> 论坛](#)。

## 参考文献

- [1] Tobias Oetiker. *A (Not So) Short Introduction to LaTeX2ε*, 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*

---

<sup>7</sup>此译本首发于 C<sup>T</sup>E<sub>X</sub> 论坛，但是需要注册才能看见链接，所以请读者自行搜索。

- Computer Typesetting*). Addison-Wesley, 2nd edition, 2004. URL <http://www.amazon.com/exec/obidos/tg/detail/-/0201362996/>.
- [3] 李果正. 大家来学 $LaTeX$ , 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] 中国TeX用户组. CTeX常见问题集, 2005. URL <http://www.ctex.org/CTEXFAQ/>.



## 第二章 入门

### 2.1 Hello, World!

把下面例子用编辑器保存为 `hello_world.tex`，这就是一个最简单的  $\text{\LaTeX}$  源文件。

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

有了源文件，我们可以在命令行把它编译成 DVI 文件（DVI 格式见[2.2.1](#)小节）。此命令知道输入的是  $\text{\LaTeX}$  源文件，所以这里的 `.tex` 后缀可以省略。以后的示例中可以省略的后缀都用 `()` 标出，不再特别声明。

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

如果系统显示类似下面的错误信息，请检查源文件是否有拼写错误。`.log` 文件里有更详细的编译信息。

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

如果编译成功，系统会报出类似下面的信息：

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

每种 L<sup>A</sup>T<sub>E</sub>X 发行包附带不同的 DVI 浏览器，比如 MiKTeX 的是 yap。

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

## 2.2 格式及其转换

### 2.2.1 页面描述语言

页面描述语言（Page Description Language, PDL）是一种在较高层次上描述实际输出结果的语言。本文只讨论其中三种与 L<sup>A</sup>T<sub>E</sub>X 紧密相关的格式：DVI、PostScript、PDF。

#### PostScript

最早的打印机只用于打印字符，它使用的硬字符与打字机类似。后来出现的点阵（dot matrix）打字机用一系列的点来“画”出字符，当然它也可以画出图形。当时矢量图的打印只能由绘图仪（plotter）来完成。

1976 年，施乐（Xerox）推出了首台激光打印机，它结合了点阵打印机和绘图仪的优点，可以同时打印高质量的图形和文字。

同一时期，John Warnock 也在酝酿一种类似于 Forth 的图形设计语言，也就是后来的 PostScript（PS），当时他正在旧金山一家电脑图形公司 Evans & Sutherland 工作。1978 年老板想让 Warnock 搬到位于犹他州的总部，他不想搬家就跳槽到了施乐。

Warnock 和 Martin Newell 开发了新的图形系统 JaM（John and Martin），它后来被合并到施乐的打印机驱动程序 InterPress 中去。这两位还开发过另一个系统 MaJ。

1982 年，Warnock 和施乐研究中心图形实验室主任 Chuck Geschke 一起离开施乐，成立了 Adobe 公司。Newell 后来也加入了 Adobe。

1984 年 Adobe 发布 PS 后不久，Steve Jobs 跑来参观，并建议用它来驱动激光打印机。次年，武装着 PS 驱动的 Apple LaserWriter 横空出世，打响了 80 年代中期桌面出版革命的第一枪。

90 年代中后期，廉价喷墨打印机的流行使得 PS 逐渐式微，因为 PS 驱动对它们毕竟是一个成本负担。

## PDF

1993 年，Adobe 推出了一种开放的格式：Portable Document Format (PDF)，它于 2007 年成为 ISO 32000 标准。除了开放，PDF 比起 PS 还有一些其它优势：

- PDF 基本上是 PS 的一个子集，因此更轻便。
- PDF 可以嵌入更先进的字体，具体见 8.2 节。
- PDF 支持嵌入乱七八糟的东东，比如动画。
- PDF 支持透明图形。

PDF 虽然拥有上述优势，起初它的推广却并不顺利，因为其读写工具 Acrobat 太贵。Adobe 很快推出了免费的 Acrobat Reader（后更名为 Adobe Reader），并不断改进 PDF，终于使它超越了曾经的事实标准 PS，成为网络时代电子文档的新标准。

## DVI

Knuth 最初设计的  $\text{T}_{\text{E}}\text{X}$  只能用于 XGP 打印机，这台打印机本身还需要一台 PDP-6 主机为它服务。1979 年，David Fuchs<sup>1</sup>提出把  $\text{T}_{\text{E}}\text{X}$  的输出改为设备无关的格式，也就是 Device Independent format (DVI)。

DVI 只是一种中间格式，用户还需要另外的处理程序 (driver) 把它转换为其它格式，比如 PS 或 PDF，甚至 PNG、SVG 等。DVI 不能嵌入字体和图形，PS 和 PDF 可以选择是否嵌入字体。

---

<sup>1</sup>Fuchs 本科毕业于普林斯顿，1978 年进入斯坦福攻读博士学位。他不是 Knuth 的学生，但是完成过一些  $\text{T}_{\text{E}}\text{X}$  的开发任务。他在 Adobe 工作过一段时间，现在混入了娱乐圈，担任过电影《Red Diaper Baby》和《Haiku Tunnel》的制片人。

## Ghostscript

PS 输出时需要一个解释器（Raster Image Processor, RIP）来把它转换为点阵图形。RIP 可以是软件，也可以是固件（firmware）或硬件<sup>2</sup>。

Ghostscript 是一个基于 RIP 的软件包，除了 RIP 它还有一些其它功能，比如处理 EPS，把 PS 转换为 PDF 等。Ghostscript 已经被移植到 Windows、Unix/Linux、Mac OS 等多种操作系统，和它匹配的前端图形用户界面（GUI）有 [GSview](#)、[Ghostview](#)、[gv](#) 等。

### 2.2.2 格式转换

DVI、PS、PDF 等格式的转换关系如图 2.1 所示。

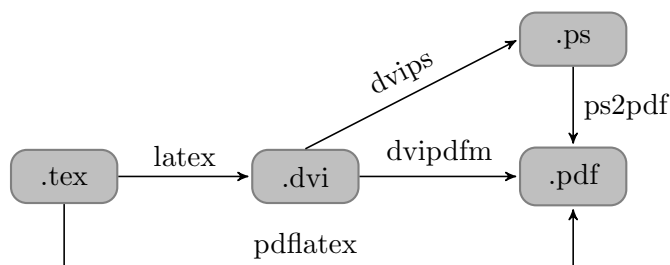


图 2.1: 格式转换

最早的 driver 是 `dvips`，它把 DVI 转换为 PS。`dvipdf` 把 DVI 转为 PDF，它后来被 `dvipdfm` 所取代；`dvipdfm` 主要用于处理单字节字符，1999 年之后停止开发；在 `dvipdfm` 基础上发展来的 `dvipdfmx` 可以处理多字节编码（字符编码详见 7.1 节）。

~~pdfTeX 是一种特殊的 driver，它跳过 DVI，直接用 TeX 源文件生成 PDF。基于 pdfTeX 的 pdfL<sup>A</sup>TeX 则把 L<sup>A</sup>TeX 源文件转为 PDF。~~

包老师倾向于 `dvipdfmx`，因为它对图形格式的兼容性较好，而且擅长处理中文。

得到 DVI 后，我们可以在控制台用以下命令把它转为 PDF。

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

<sup>2</sup>固件 RIP 在打印机内置处理器上运行，硬件 RIP 常见于高端打印设备。

我们也可以把它转为 PS，接着用 Ghostscript 的一个命令行程序把它转换为 PDF，注意第二个命令需要 .ps 后缀。一般情况下不推荐这种方法，因为它多了个步骤。

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

pdfL<sup>A</sup>T<sub>E</sub>X 用法如下。

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

## 2.3 L<sup>A</sup>T<sub>E</sub>X 语句

L<sup>A</sup>T<sub>E</sub>X 源文件的每一行称作一条语句（statement），语句可以分三种：命令（command）、数据（data）和注释（comment）。

命令分为两种：普通命令和环境（environment）。普通命令以\起始，大多只有一行；而环境包含一对起始声明和结尾声明，用于多行的场合。命令和环境可以互相嵌套。

数据就是普通内容。注释语句以%起始，它在编译过程中被忽略。

例如在2.1节例1中，第一行是注释，第二行是普通命令；第三、五行是环境的起始和结尾声明；第四行是数据。

## 2.4 文档结构

### 2.4.1 文档类、序言、正文

L<sup>A</sup>T<sub>E</sub>X 源文件的结构分三大部分，依次为：文档类声明、序言（可选）、正文。

文档类声明用来指定文档的类型；序言（preamble）用来完成一些特殊任务，比如引入宏包，定义命令，设置环境等；文档的实际内容则放在正文部分。这里的正文指得是\begin{document}和\end{document}之间的部分，和通常人们心目中的“正文”概念有所出入。

这三部分的基本语法如下：

```

\documentclass[options]{class} %文档类声明
\usepackage[options]{package} %引入宏包
...
\begin{document}              %正文
...
\end{document}

```

常用的文档类（documentclass）有三种：`article`、`report`、`book`，它们的常用选项见表 2.1。

表 2.1: 文档类常用选项

10pt, 11pt, 12pt	正文字号，缺省10pt。L <sup>A</sup> T <sub>E</sub> X 会根据正文字号选择标题、上下标等的字号。
letterpaper, a4paper	纸张尺寸，缺省是 letter。
notitlepage, titlepage	标题后是否另起新页。article 缺省 notitlepage, report 和 book 缺省有 titlepage。
onecolumn, twocolumn	栏数，缺省单栏。
oneside, twoside	单面双面。article 和 report 缺省单面，book 缺省双面。
landscape	打印方向横向，缺省纵向。
openany, openright	此选项只用于 report 和 book。report 缺省 openany，book 缺省 openright。
draft	草稿模式。有时某些行排得过满，draft 模式可以在它们右边标上粗黑线提醒用户。

L<sup>A</sup>T<sub>E</sub>X 的核心只提供基本的功能，系统以宏包（package）的形式提供附加功能或增强原有功能。其它一些编程语言也有类似的模块化机制，比如 C/C++ 的 `#include`，Java 的 `import`。

### 2.4.2 标题、摘要、章节

一份文档正文部分的开头通常有标题、作者、摘要等信息，之后是章节等层次结构，内容则散布于层次结构之间。

标题、作者、日期等命令如下，注意`\maketitle`命令要放在最后。

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

摘要环境用法如下：

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

常用的层次结构命令如下，

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

每个高级层次可以包含若干低级层次。`article` 中没有 `chapter`，而 `report` 和 `book` 则支持上面所有层次。

### 2.4.3 目录

我们可以用 `\tableofcontents` 命令来生成整个文档的目录，`LATEX` 会自动设定目录包含的章节层次，也可以用 `\setcounter` 命令来指定目录层次深度。

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

如果不想让某个章节标题出现在目录中，可以使用以下带 `*` 的命令来声明章节。

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

类似地，我们也可以用以下命令生成插图和表格的目录，插图和表格功能将在后面章节中介绍。

```
\listoffigures
\listoftables
```

当章节或图表等结构发生变化时，我们需要执行两遍编译命令以获得正确结果。L<sup>A</sup>T<sub>E</sub>X 之所以设计成这样可能是因为当时的电脑内存容量有限。

## 2.5 文字排版

### 2.5.1 字符输入

文档中可以输入的内容大致可以分为：普通字符、控制符、特殊符号、注音符号、预定义字符串等。而这些内容有两种输入模式：文本模式（缺省）和数学模式，普通的行间（inline）数学模式用 $\$...\$$ 来表示。

L<sup>A</sup>T<sub>E</sub>X 中有些字符（例如 # \$ % ^ & \_ { } ~ \ 等）被用作特殊的控制符，所以不能直接输入，多数需要在前面加个 \。而 \ 本身则要用 `\textbackslash` 命令来输入，因为 `\\` 被用作了换行指令。很奇怪为什么不用 C 语言的 `\n`，也许是因为 T<sub>E</sub>X 的编程语言是 Pascal。

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

表 2.2 提供了一些符号的输入方法示例，完整的符号列表见 Scott Pakin 的《The Comprehensive L<sup>A</sup>T<sub>E</sub>X Symbol List》<sup>[1]</sup>。

### 2.5.2 换行、换页、断字

通常 L<sup>A</sup>T<sub>E</sub>X 会自动换行、换页。用户也可以用 `\\` 或 `\newline` 来强制换行；用 `\newpage` 来强制换页。

一般情况下 L<sup>A</sup>T<sub>E</sub>X 会尽量均匀地断字（Hyphenate），使得每一行的字间距分布整齐。但有时我们也需要显式指明断字位置，比如下例就指明 BASIC 这个词不能断开，而 blar-blar-blar 可以在 - 处断开。

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



表 2.2: 一些符号和预定义字符串

特殊符号	注音符	预定义字符串
© <code>\textcopyright</code>	å <code>\aa</code>	August 26, 2008 <code>\today</code>
® <code>\textregistered</code>	Å <code>\AA</code>	T <sub>E</sub> X <code>\TeX</code>
°C <code>\$^\circ\$C</code>	æ <code>\ae</code>	L <sup>A</sup> T <sub>E</sub> X <code>\LaTeX</code>
¥ <code>\textyen</code>	ø <code>\o</code>	L <sup>A</sup> T <sub>E</sub> X 2 <sub>ε</sub> <code>\LaTeXe</code>
£ <code>\pounds</code>	ö <code>\"o</code>	METAFONT <code>\MF</code>
€ <code>\texteuro</code>	ô <code>\~o</code>	METAPOST <code>\MP</code>
... <code>\dots</code>	õ <code>\~o</code>	

### 2.5.3 字样、字号

L<sup>A</sup>T<sub>E</sub>X 会自动调整正文、标题、章节、上下标、脚注等的字样<sup>3</sup>、字号。我们也可以用表 2.3 中的命令来设置字样；用表 2.4 中的命令来设置相对字号，比如正文字号是 10pt、11pt、12pt 时，tiny 的字号就分别是 5pt、6pt、6pt。

L<sup>A</sup>T<sub>E</sub>X 有一个特别的字样强调命令：`\emph`，它在不同字样和装饰环境下有不同效果。比如周围文字是正体，它就是斜体；反之它就是正体。

表 2.3: 字样命令

<code>\textrm{...}</code>	roman	<code>\textbf{...}</code>	<b>bold face</b>
<code>\textsf{...}</code>	sans serif	<code>\textit{...}</code>	<i>italic</i>
<code>\texttt{...}</code>	typewriter	<code>\textsl{...}</code>	<i>slanted</i>
<code>\emph{...}</code>	<i>emphasized</i>	<code>\underline{...}</code>	<u>underline</u>
<code>\textsc{...}</code>	SMALL CAPS		

<sup>3</sup>关于字样详见 8.1 节

表 2.4: 字号命令

命令	正文字号		
	10pt	11pt	12pt
<code>\tiny</code>	5pt	6pt	6pt
<code>\scriptsize</code>	7pt	8pt	8pt
<code>\footnotesize</code>	8pt	9pt	10pt
<code>\small</code>	9pt	10pt	11pt
<code>\normalsize</code>	10pt	11pt	12pt
<code>\large</code>	12pt	12pt	14pt
<code>\Large</code>	14pt	14pt	17pt
<code>\LARGE</code>	17pt	17pt	20pt
<code>\huge</code>	20pt	20pt	25pt
<code>\Huge</code>	25pt	25pt	25pt

## 2.6 常用命令环境

### 2.6.1 列表

L<sup>A</sup>T<sub>E</sub>X 中有三种列表环境：`itemize`、`enumerate`、`description`，它们的一般用法如下：

```
\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++} 一种编程语言
  \item{Java} 另一种编程语言
  \item{HTML} 一种标记语言
\end{description}
```

```
C++ 一种编程语言
Java 另一种编程语言
HTML 一种标记语言
```

### 2.6.2 对齐

L<sup>A</sup>T<sub>E</sub>X 中的段落缺省两端对齐（fully justified），我们也可以让段落居左、居右或居中对齐。

```
\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 摘录

L<sup>A</sup>T<sub>E</sub>X 中有三种摘录环境：`quote`、`quotation`、`verse`。`quote` 两端都缩进，`quotation` 在 `quote` 的基础上增加了首行缩进，`verse` 比 `quote` 多了第二行起的缩进。

```
正文
\begin{quote}
引文两端都缩进。
\end{quote}
正文
```

```
正文
      引文两端都缩进。
正文
```

```
正文
\begin{quotation}
引文两端缩进，首行缩进。
\end{quotation}
正文
```

```
正文
      引文两端缩进，首行缩进。
正文
```

```
正文
\begin{verse}
引文两端缩进，第二行起缩进。
\end{verse}
正文
```

```
正文
      引文两端缩进，
      第二行起缩进。
正文
```

#### 2.6.4 原文照排

一般文档中，命令和源代码通常使用等宽字样来表示，也就是原文照排。对此 L<sup>A</sup>T<sub>E</sub>X 提供了 `\verb` 命令（一般用于在正文中插入较短的命令）和 `verbatim` 环境。后者有带 `*` 的版本用来标明空格。

```
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

我们常常需要引用文档中 `section`、`subsection`、`figure`、`table` 等对象的编号，这种功能叫作交叉引用（cross referencing）。

L<sup>A</sup>T<sub>E</sub>X 中可以用 `\label{marker}` 命令来定义一个标记，标记名可以是任意字符串，但是在全文中须保持唯一。之后可以用 `\ref{marker}` 命令来引用标记处章节或图表的编号，用 `\pageref{marker}` 来引用标记处的页码。

被引用处`\label{sec}``\`  
`\`  
 第`\pageref{sec}`页`\ref{sec}`节

被引用处  
 ...  
 第19页2.6.5节

文档中新增交叉引用后，第一次执行 `latex` 或 `pdflatex` 编译命令时会得到类似下面的警告信息。因为第一次编译只会扫描出有交叉引用的地方，第二次编译才能得到正确结果。

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

### 2.6.6 脚注

脚注（footnote）的一般用法如下：

这里是一段正文。`\footnote{这里是一段脚注。}`

这里是一段正文。<sup>a</sup>  
<sup>a</sup>这里是一段脚注。

## 2.7 长度单位

L<sup>A</sup>T<sub>E</sub>X 中的常用长度单位如表 2.5 所示。point 是个传统印刷业采用的单位，而 big point 是 Adobe 推出 PS 时新定义的单位。em 是个相对单位，比如当前字体是 11pt 时，1em 就是 11pt。

表 2.5: 常用长度单位

in	英寸	pt	point, 1/72.27 in	em	当前字体中字母M的宽度
cm	厘米	bp	big point, 1/72 in	ex	当前字体中字母x的高度
mm	毫米	pc	pica, 12 pt	mu	math unit, 1/18 em

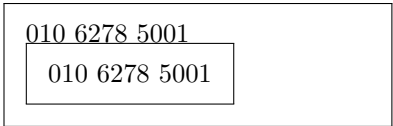
## 2.8 盒子

L<sup>A</sup>T<sub>E</sub>X 在排版时把每个对象（小到一个字母，大到一个段落）都视为一个矩形盒子（box），我们在 HTML 和 CSS 中也可以见到类似的模型。

### 2.8.1 mbox 和 fbox

L<sup>A</sup>T<sub>E</sub>X 中最简单的盒子是 `\mbox` 和 `\fbox`。前者把一组对象组合起来，后者在此基础上加了个边框。


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



### 2.8.2 makebox 和 framebox

稍复杂的 `\makebox` 和 `\framebox` 提供了宽度和对齐方式控制选项。这里用 l、r、s 分别代表居左、居右和分散对齐。

```
%语法: [宽度][对齐方式]{内容}
\makebox[100pt][l]{居左}
\framebox[100pt][r]{居右}
```



### 2.8.3 parbox 和 minipage

大一些的对象比如整个段落可以用 `\parbox` 命令和 `\minipage` 环境，两者语法类似，也提供了对齐方式和宽度的选项。但是这里的对齐方式是指与周围内容的纵向关系，用 t、c、b 分别代表居顶、居中和居底对齐。

```
%语法: [对齐方式]{宽度}{内容}  
\parbox[c]{90pt}{锦 瑟 无 端 五 十  
弦, \一弦一柱思华年。}李商隐
```

锦瑟无端五十弦, 一弦一柱思华年。	李商隐
----------------------	-----

细心的读者会发现 `\parbox` 和 `\minipage` 的选项排列顺序和 `\makebox` 和 `\framebox` 的不一致, 可能出自不同的作者。

## 参考文献

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

看什么看，没见过空白页？



## 第三章 数学

今有上禾三秉，中禾二秉，下禾一秉，实三十九斗；上禾二秉，中禾三秉，下禾一秉，实三十四斗；上禾一秉，中禾二秉，下禾三秉，实二十六斗。问上、中、下禾实一秉各几何？

$$3x + 2y + z = 39$$

$$2x + 3y + z = 34$$

$$x + 2y + 3z = 26$$

— 《九章算术》

为了使用  $\mathcal{A}\mathcal{M}\mathcal{S}$ - $\text{\LaTeX}$  提供的数学功能，我们首先需要在文档的序言部分加载 `amsmath` 宏包，其详细用法可参阅《`amsmath` User's Guide》<sup>[1]</sup>。更全面的数学内容排版可参阅 George Grätzer<sup>1</sup>的《More Math into  $\text{\LaTeX}$ , 4th Edition》<sup>[2]</sup>。

### 3.1 数学模式

$\text{\LaTeX}$  中的数学模式有两种形式：`inline` 和 `display`。前者是指在正文插入行间数学公式，后者独立排列，可以有或没有编号。

行间公式用一对 `$...$` 来输入，独立公式用 `equation` 或 `equation*` 环境来输入，有 `*` 的版本不生成公式编号。

前文提到 `\fbox` 命令可以给文本内容加个方框，数学模式下也有类似的命令 `\boxed`。

---

<sup>1</sup>匈牙利裔，加拿大 Manitoba 大学数学系教授。

爱因斯坦的 $E=mc^2$ 方程

```
\begin{equation}
E=mc^2
\end{equation}
\[ E=mc^2 \]
\[ \boxed{E=mc^2} \]
```

爱因斯坦的 $E = mc^2$ 方程

$$E = mc^2 \quad (3.1)$$

$$E = mc^2$$

$$E = mc^2$$

## 3.2 基本原素

### 3.2.1 字母

英文字母在数学模式下可以直接输入，希腊字母则需要用表 3.1 中的命令输入，注意大写希腊字母的命令首字母也是大写。

表 3.1: 希腊字母

$\alpha$	<code>\alpha</code>	$\theta$	<code>\theta</code>	$o$	<code>o</code>	$\tau$	<code>\tau</code>
$\beta$	<code>\beta</code>	$\vartheta$	<code>\vartheta</code>	$\pi$	<code>\pi</code>	$\upsilon$	<code>\upsilon</code>
$\gamma$	<code>\gamma</code>	$\iota$	<code>\iota</code>	$\varpi$	<code>\varpi</code>	$\phi$	<code>\phi</code>
$\delta$	<code>\delta</code>	$\kappa$	<code>\kappa</code>	$\rho$	<code>\rho</code>	$\varphi$	<code>\varphi</code>
$\epsilon$	<code>\epsilon</code>	$\lambda$	<code>\lambda</code>	$\varrho$	<code>\varrho</code>	$\chi$	<code>\chi</code>
$\varepsilon$	<code>\varepsilon</code>	$\mu$	<code>\mu</code>	$\sigma$	<code>\sigma</code>	$\psi$	<code>\psi</code>
$\zeta$	<code>\zeta</code>	$\nu$	<code>\nu</code>	$\varsigma$	<code>\varsigma</code>	$\omega$	<code>\omega</code>
$\eta$	<code>\eta</code>	$\xi$	<code>\xi</code>				
$\Gamma$	<code>\Gamma</code>	$\Lambda$	<code>\Lambda</code>	$\Sigma$	<code>\Sigma</code>	$\Psi$	<code>\Psi</code>
$\Delta$	<code>\Delta</code>	$\Xi$	<code>\Xi</code>	$\Upsilon$	<code>\Upsilon</code>	$\Omega$	<code>\Omega</code>
$\Theta$	<code>\Theta</code>	$\Pi$	<code>\Pi</code>	$\Phi$	<code>\Phi</code>		

### 3.2.2 指数、下标、根号

指数或上标用 `^` 表示，下标用 `_` 表示，根号用 `\sqrt` 表示。上下标如果多于一个字母或符号，需要用一对 `{}` 括起来。

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

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

### 3.2.3 分数

分数用 `\frac` 命令表示，它会自动调整字号，比如在行间公式中小一点，在独立公式则大一点。`\dfrac` 命令把分数的字号显式设置为独立公式中的大小，`\tfrac` 命令则把字号设为行间公式中的大小。

```


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


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


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


```

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

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

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

### 3.2.4 运算符

有些小的运算符 (operator) 例如 `+` `-` `*` `/` 等可以直接输入，另一些则需要特殊命令。完整的数学符号参见 Scott Pakin 的《The Comprehensive L<sup>A</sup>T<sub>E</sub>X Symbol List》[\[3\]](#)。

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

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

和、积、极限、积分等大运算符用 `\sum` `\prod` `\lim` `\int` 等表示。它们的上下标在行间公式中被压缩，以适应行高。。

```


$$\sum_{i=1}^n i \quad \prod_{i=1}^n i \quad \lim_{x \rightarrow 0} x^2 \quad \int_a^b x^2 dx$$


$$\sum_{i=1}^n i \quad \prod_{i=1}^n i \quad \lim_{x \rightarrow 0} x^2 \quad \int_a^b x^2 dx$$


```

$$\sum_{i=1}^n i \quad \prod_{i=1}^n i \quad \lim_{x \rightarrow 0} x^2 \quad \int_a^b x^2 dx$$

$$\sum_{i=1}^n i \quad \prod_{i=1}^n i \quad \lim_{x \rightarrow 0} x^2 \quad \int_a^b x^2 dx$$

多重积分如果用多个 `\int` 来输入的话，积分号间距过宽。正确的方法是用 `\iint` `\iiint` `\iiiint` `\idotsint` 等命令输入。从下例中可以看出两种方法的差异。

$$\begin{array}{cccc} \iint & \iiint & \iiiint & \int \cdots \int \\ \iint & \iiint & \iiint & \int \cdots \int \end{array}$$

### 3.2.5 分隔符

各种括号用 `()` `[]` `\{\}` `\langle\rangle` 等命令表示，注意花括号通常用来输入命令和环境的参数，所以在数学公式中它们前面要加 `\`。因为  $\text{\LaTeX}$  中的 `|` 和 `\|` 的应用过于随意，`amsmath` 宏包推荐用 `\lvert\rvert` 和 `\lVert\rVert` 取而代之。

我们可以在上述分隔符前面加 `\big` `\Big` `\bigg` `\Bigg` 等命令来调整大小。 $\text{\LaTeX}$  原有的方法是在分隔符前面加 `\left` `\right` 来自动调整大小，但是效果不佳，所以 `amsmath` 不推荐用这种方法。

$$\begin{array}{ccc} \left(\left(\left(\left(x+y\right)\right)\right)\right) & \left[\left[\left[x+y\right]\right]\right] & \left\{\left\{\left\{x+y\right\}\right\}\right\} \\ \left\langle\left\langle\left\langle x+y\right\rangle\right\rangle\right\rangle & \left\|x+y\right\| & \left\|x+y\right\| \end{array}$$

### 3.2.6 箭头

表 3.2 列出了部分箭头的输入方法。另外还有两个命令生成的箭头可以根据上下标自动调整长度。

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

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

表 3.2: 箭头

$\leftarrow$	<code>\leftarrow</code>	$\longleftarrow$	<code>\longleftarrow</code>
$\rightarrow$	<code>\rightarrow</code>	$\longrightarrow$	<code>\longrightarrow</code>
$\leftrightarrow$	<code>\leftrightarrow</code>	$\longleftrightarrow$	<code>\longleftrightarrow</code>
$\Lleftarrow$	<code>\Lleftarrow</code>	$\Longleftarrow$	<code>\Longleftarrow</code>
$\Rrightarrow$	<code>\Rrightarrow</code>	$\Longrightarrow$	<code>\Longrightarrow</code>
$\Leftrightarrow$	<code>\Leftrightarrow</code>	$\Longleftrightarrow$	<code>\Longleftrightarrow</code>

### 3.2.7 标注

表 3.3 列出一些短的注音符号（accent），表 3.4 则列出一些长的标注符号。

表 3.3: 数学注音符号

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

表 3.4: 长标注符号

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

### 3.2.8 省略号

省略号用 `\dots` `\cdots` `\vdots` `\ddots` 等命令表示，注意 `\cdots` 和 `\dots` 的差别。

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

### 3.2.9 空白间距

在数学模式中，我们可以用表 3.5 中的命令生成不同的间距，注意负间距命令 `\!` 可以用来减小间距。

表 3.5: 空白间距

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

## 3.3 矩阵和行列式

数学模式下可以用 `array` 环境来生成行列表。参数 `{ccc}` 用于设置每列的对齐方式，`l`、`c`、`r` 分别表示左中右；`\\` 和 `&` 用来分隔行和列。

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

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

`amsmath` 有几个类似的环境：`pmatrix`、`bmatrix`、`Bmatrix`、`vmatrix` 和 `Vmatrix`，它们和 `array` 的主要区别是会在表两端加上 `()` `[]` `{}` `||` `|||` 等分隔符，其次这些环境没有列对齐方式参数。

行间公式可以用 `smallmatrix` 环境来生成排列紧密的小矩阵。

## 3.4 多行公式

有时一个公式太长一行放不下，或几个公式需要写成一组，这时我们就要用到 `amsmath` 提供的几个适合多行公式的环境。

### 3.4.1 长公式

对于多行不需要对齐的长公式，我们可以用 `multline` 环境。

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

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

需要对齐的长公式可以用 `split` 环境，它本身不能单独使用，因此也称作次环境，必须包含在 `equation` 或其它数学环境内。`split` 环境用 `\\` 和 `&` 来分行和设置对齐位置。

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

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

### 3.4.2 公式组

不需要对齐的公式组用 `gather` 环境，需要对齐的用 `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}
```

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

`multline`、`gather`、`align` 等环境都有带 `*` 的版本，它们不生成公式编号。

有多种条件的公式组用 `cases` 次环境。

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

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

### 3.5 定理和证明

LaTeX 提供了一个 `\newtheorem` 命令来定义定理之类的环境，其语法如下。

```
\newtheorem{环境名}[编号延续]{显示名}[编号层次]
```

在下例中，我们定义了四个环境：定义、定理、引理和推论，它们都在一个 `section` 内编号，而引理和推论会延续定理的编号。

```
\newtheorem{defination}{定义}[section]
\newtheorem{theorem}{定理}[section]
\newtheorem{lemma}[theorem]{引理}
\newtheorem{corollary}[theorem]{推论}
```

定义了上述环境之后，我们就可以象下面这样使用它们。

```
\begin{defination}
Java是一种跨平台的编程语言。
\end{defination}
```

**定义3.5.1.** *Java*是一种跨平台的编程语言。

```
\begin{theorem}
咖啡因会使人的大脑兴奋。
\end{theorem}
```

**定理3.5.1.** 咖啡因会使人的大脑兴奋。

```
\begin{lemma}
茶和咖啡都会使人兴奋。
\end{lemma}
```

**引理3.5.2.** 茶和咖啡都会使人兴奋。



```
\begin{corollary}
晚上喝咖啡会导致失眠。
\end{corollary}
```

**推论3.5.3.** 晚上喝咖啡会导致失眠。

`proof`环境可以用来输入下面这样的证明，它会在证明结尾输入一个QED符号<sup>2</sup>。

```
\begin{proof}[命题“物质无限可分”的
证明]
一尺之棰，日取其半，万世不竭。
\end{proof}
```

命题“物质无限可分”的证明。  
一尺之棰，日取其半，万世不竭。  $\square$

## 3.6 数学字体

和文本模式类似，数学模式下可以用表 3.6 中的命令选择不同字体，其中有些字体需要加载 `amsfonts` 宏包。

表 3.6: 数学字体

缺省	<code>\mathbf</code>	<code>\mathit</code>	<code>\mathsf</code>	<code>\mathcal</code>	<code>\mathbb</code>
XNZRC	<b>XNZRC</b>	<i>XNZRC</i>	XNZRC	$\mathcal{XNZRC}$	$\mathbb{XNZRC}$

## 参考文献

- [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/>.

<sup>2</sup>拉丁语 quod erat demonstrandum 的缩写。

看什么看，没见过空白页？

## 第四章 插图

A picture says more than a thousand words.

— Shakespeare

当年 Knuth 开发  $\text{T}_{\text{E}}\text{X}$  时，GIF、JPEG、PNG、EPS 等图形格式还没有问世，所以 DVI 不能直接支持这些格式。但是高手就是高手，Knuth 在  $\text{T}_{\text{E}}\text{X}$  上留了一个后门：`\special` 命令，让后面的 Driver 决定怎样处理图形。

这和当年老毛把港澳台，老邓把钓鱼岛都“留给后人解决”有异曲同工之妙。曾经有位出版社的编辑看上了包老师写的一个程序，要包老师改改当作教学辅助软件出版，但是包老师手头没有 DOS 中断的资料没办法加鼠标操作。该编辑说：你把鼠标驱动打包在软件里，让用户自己琢磨是怎么回事。

下面我们会在 4.1 节介绍一下  $\text{L}^{\text{A}}\text{T}_{\text{E}}\text{X}$  所用图形格式，4.2 节介绍怎样插入已有的图形，4.4–4.6 节讨论怎样制作矢量图形。

### 4.1 图形格式

$\text{L}^{\text{A}}\text{T}_{\text{E}}\text{X}$  支持点阵图形格式 JPEG 和 PNG，也支持矢量格式 EPS 和 PDF。对于示意图，我们应该首选矢量格式；包含大量自然色彩的图像（比如照片）应该选 JPEG，人工点阵图像应该选 PNG。

### 4.1.1 EPS

80 年代中后期，PS 风头之劲一时无两，人们自然会考虑把它作为文档中嵌入图形的标准格式。然而 PS 实在太强大，人们担心嵌入文档的 PS 会搞破坏，于是就产生了戴着手铐的 Encapsulated PostScript (EPS)。出于同样的原因，人们也担心嵌入 HTML 的 ActiveX、Java Applet、JavaScript 中混入恶意代码，所以才会对它们也有所限制。

早年间 DVI 经常被转换为 PS，所以 EPS 就成了  $\text{\LaTeX}$  的标准图形格式。

### 4.1.2 Driver 们的口味

#### dvips

dvips 喜欢 PS，所以就爱屋及乌只支持嵌入 EPS。MiKTeX 看不惯这种垄断行为，就把 dvips 破解，添加了对 JPEG 和 PNG 的支持。如果你反对这种黑客破解行径，只好找软件把其它图形格式转换为 EPS。

#### pdf $\text{\LaTeX}$

pdf $\text{\LaTeX}$  支持 JPEG、PNG 和 PDF，不支持 EPS。传说 pdf $\text{\LaTeX}$  不支持 EPS 的原因是 PS 解释器的版权问题。包老师认为这种说法不可信，因为 1997 年 Hàn The Thành 发布 pdf $\text{\TeX}$  时 PS 已经被 PDF 赶超，Adobe 与其保护 PS 还不如保护 PDF。

$\text{\LaTeX}$  有两个宏包 `epstopdf` 和 `pst-pdf` 可以实时地 (on the fly) 把 EPS 转换为 PDF<sup>1</sup>。然而前者有安全漏洞，后者用法繁琐，用户最好还是用其它软件事先把 EPS 转为 PDF。

#### dvipdfm

dvipdfm 支持 JPEG、PNG、PDF，不支持 EPS，但是它可以实时地调用 Ghostscript 把 EPS 转为 PDF。所以从图形格式支持的角度来

---

<sup>1</sup>在这里 on the fly 是指在后台处理，用户不用操心。包老师不确定把它翻译为“实时”是否合适，因为 real time 通常被翻译为实时。对于用户无须干涉、知情的情况，有人说 user transparent，也有人说 black box，语言还真奇妙。

讲，`dvipdfm` 比 `dvips` 和 `pdfLATEX` 都好。

那位同学说了，你这么多废话作甚，直接告诉我们用 `dvipdfm` 不就完了。然而你别忘了 `dvipdfm` 需要 DVI 作为输入，不幸的是用来生成 DVI 的 `latex` 对 JPEG 和 PNG 有意见。

综上所述，这些 driver 都不能把图形格式痛快地通吃，所以同学们别着急，且听包老师慢慢忽悠怎样转换和处理图形格式。

### 4.1.3 图形格式转换

注意把点阵图形转换为矢量图形并不能提高图形本身的质量，正所谓“garbage in, garbage out”。

#### JPEG 和 PNG 的范围框

作为中间格式的 DVI 不包含图形本身，它只记录图形的尺寸和文件名，因为具体的图形处理由后面的 driver 负责。DVI 中图形的尺寸来自它的范围框（bounding box），而 `latex` 无法从点阵图形文件中提取这一信息，所以我们需要以某种方式把范围框信息告诉它。

一种方法是打开图形文件，记下尺寸，插入图形时加上相应的参数。

另一种方法用 `ebb` 程序生成一个含范围框信息的文件。比如下例会生成 `graph.bb` 文件，有了它插入图形时就不需要范围框参数。注意有时此程序算出的范围框不准，不知道是它的 bug 还是包老师的人品问题。

```
ebb graph.jpg
```

#### 其它格式转为EPS

有很多程序都可以把点阵图形转换为 EPS，比如 [ImageMagick](#)，以及 [a2ping/sam2p](#)、[bmeps](#)、[jpeg2ps](#)、[sam2p](#) 等。

PS 从 Level 2 开始才支持点阵图形压缩，所以在把其它格式转为 EPS 时应尽量使用 Level 2 或 3，否则输出的 EPS 会很大。

下面是一个 ImageMagick 中 `convert` 程序的例子。

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

另外还有一种 PS 虚拟打印机的方法，优点是几乎可以把几乎所有文件“打印”成 EPS，缺点是输出的是 PS Level 1，即使驱动程序提供了其它 Level 的选项。

1. 找一个 PS 打印机驱动程序。Windows 安装盘附带很多打印机驱动，其中带 PS 字样的就是 PS 驱动。包老师选的是“HP Color LaserJet 8550-PS”，Adobe 提供的 PS 驱动效果不太好。其它驱动安装过程可能稍有不同。
2. 安装时端口选“FILE”，或者后面打印时选择“Print to File”。高级选项里的 PS 选项选“Encapsulated PostScript (EPS)”。
3. 打开点阵图形文件，打印到上面的虚拟打印机，输出的文件就是 EPS，但是它没有范围框。
4. 用 GSview 打开上面生成的 EPS，不用理会没范围框的警告，Options 菜单里选上“EPS Clip”，用 File 菜单的“PS to EPS”生成含范围框的 EPS。

### 其它格式转为 PDF

L<sup>A</sup>T<sub>E</sub>X 附带的 `epstopdf` 程序<sup>2</sup>可以把 EPS 转为 PDF。类似地我们也可以安装一个 PDF 虚拟打印机，用它来把其它图形文件转为 PDF。

## 4.2 插入图形

### 4.2.1 插入命令

如今万事俱备，只欠插入。上回讲到哪儿来着？Yeah，Knuth 的后门 `\special`。

用低级命令 `\special` 来插入图形很不爽，于是 L<sup>A</sup>T<sub>E</sub>X v2.09 增加了 `epsf` 和 `psfig` 宏包。之后 L<sup>A</sup>T<sub>E</sub>X 2<sub>ε</sub> 推出了更好的 `graphics` 和 `graphicx` 宏包，这两个宏包有个共同的命令：`\includegraphics`。`graphicx` 版本的语法更简单，功能更强大，所以一般推荐用它。

---

<sup>2</sup>这个命令程序和上面提到的 `epstopdf` 宏包是两样东西。

插入图形的具体命令如下，如果是点阵图形需要加范围框参数（左上角和右下角坐标）。

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

若想省略文件后缀，可在插入图形前使用两个命令。前者指定一个后缀列表，让 L<sup>A</sup>T<sub>E</sub>X 自行查找；后者告诉 L<sup>A</sup>T<sub>E</sub>X 未知后缀的都是 EPS。

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

```

### 4.2.2 缩放、旋转

表 4.1 和表 4.2 的选项可以用来缩放、旋转和裁剪插图。

表 4.1: includegraphics 命令的缩放和旋转选项

scale	缩放比例
width	宽度
height	高度
totalheight	范围框高度，旋转时它不等于高度
keepaspectratio	如果不使用它而同时指定插图的宽度、高度，长宽比可能会失调；使用它时长宽比不变，宽度、高度都不超过指定参数
angle	旋转角度
origin	旋转原点

表 4.2: includegraphics 命令的裁剪选项

viewport	可视区域的左上角和右下角坐标。
trim	左、下、右、上四边裁剪的数值。
clip	是否真正裁剪。缺省为false，不执行裁剪，多出的部分就那样放着；设置为true时执行裁剪。似乎是多此一举。

若想深入了解 L<sup>A</sup>T<sub>E</sub>X 插入图形的功能，请参考 Keith Reckdahl 的《Using Imported Graphics in L<sup>A</sup>T<sub>E</sub>X and pdfL<sup>A</sup>T<sub>E</sub>X》<sup>[1]</sup>（简称epslatex）。

### 4.2.3 figure环境

插图通常需要占据大块空白，所以在文字处理软件中用户经常需要调整插图的位置。L<sup>A</sup>T<sub>E</sub>X 有一个 **figure** 环境可以自动完成这样的任务，这种自动调整位置的环境称作浮动环境。

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



图 4.1: 10个月大的Anna



上述代码中，`[htbp]` 选项用来指定插图排版的理想位置，这几个字母分别代表 here、top、bottom、float page，也就是固定位置、页顶、页尾、单独的浮动页。我们可以使用这几个字母的任意组合，一般不推荐单独使用 `[h]`，因为那个位置也许很不合适， $\text{\LaTeX}$  会很生气。

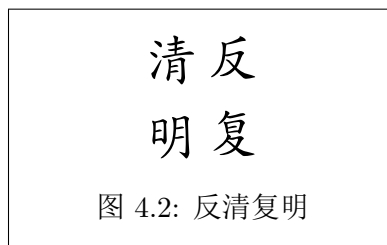
`\centering` 用来使插图居中，`\caption` 命令设置插图标题， $\text{\LaTeX}$  会自动给浮动环境的标题加上编号。注意 `label` 应放在 `caption` 之后，否则引用时指向的是前一个插图。

#### 4.2.4 插入多幅图形

##### 并排摆放，共享标题

当我们需要两幅图片并排摆放，并共享标题时，可以在 `figure` 环境中使用两个 `\includegraphics` 命令。

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



##### 并排摆放，各有标题

如果想要两幅并排的图片各有自己的标题，可以在 `figure` 环境中使用两个 `minipage` 环境，每个环境里插入一个图。

```
\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{反复}
\end{minipage}
\end{figure}

```

清  
明

图 4.3: 清明

反  
复

图 4.4: 反复

### 并排摆放，共享标题，各有子标题

如果想要两幅并排的图片共享一个标题，并各有自己的子标题，可以使用 `subfig` 宏包提供的 `\subfloat` 命令。

`subfloat` 命令缺少宽度参数。虽然我们可以用 `\hspace` 命令调整子图的距离，子标题却只能和子图本身一样宽，就会出现折行。

```

\usepackage{subfig}
\begin{figure}[htbp]
  \centering
  \subfloat[清明]{
    \label{fig:subfig_a}
    \includegraphics{left}
  }
  \hspace{80pt}
  \subfloat[反复]{
    \label{fig:subfig_b}
    \includegraphics{right}
  }
  \caption{反清复明}
\end{figure}

```

每个子图可以有各自的引用，就象这个样子：图 4.5a、图 4.5b。



图 4.5: 反清复明

### 改进的子图方法

为了避免子标题折行，我们可以在 `\subfloat` 里再嵌套个 `minipage`，因为后者是有宽度的。

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



图 4.6: 反清复明

### 4.3 图形绘制工具比较

与  $\text{\LaTeX}$  配套使用的绘图工具主要有三种： $\text{METAPOST}$ 、 $\text{PSTricks}$  和  $\text{PGF}$ ，它们的特点如下。

- 工作方式。 $\text{METAPOST}$  离线绘图，生成的 EPS 可以插入  $\text{\LaTeX}$  文档； $\text{PSTricks}$  和  $\text{PGF}$  都采用在线绘图的方式，也就是  $\text{\LaTeX}$  文档内直接使用绘图命令。
- 兼容性。 $\text{METAPOST}$  生成的 MPS 需要先转为 PDF 才能被  $\text{pdf\LaTeX}$  使用； $\text{PSTricks}$  生成的 EPS 和  $\text{pdf\LaTeX}$  不兼容； $\text{PGF}$  提供针对各种 driver 的接口，兼容性最好。
- 功能。 $\text{PSTricks}$  有 PS 作后盾，功能最强； $\text{METAPOST}$  擅长处理数学内容； $\text{PGF}$  的流程图有独到之处。

限于篇幅，本文只对这三种工具进行简介。除了它们，用户也可以考虑一些面向  $\text{\LaTeX}$  的绘图前端，比如 Unix/Linux 下的  $\text{xfig}$  和 Windows 下的  $\text{TpX}$ ；或可以输出 EPS 的专用软件，比如  $\text{gnuplot}$  和  $\text{Matlab}$ 。

### 4.4 $\text{METAPOST}$

1989 年 John D. Hobby<sup>3</sup>开始设计一种绘图语言及其编译器，也就是  $\text{METAPOST}$ 。 $\text{METAPOST}$  从  $\text{METAFONT}$  那里获得了大量灵感和源代码，学生从导师那里顺点东西自然是手到擒来。 $\text{METAPOST}$  和  $\text{METAFONT}$  语

<sup>3</sup>Hobby 1985 年从斯坦福获博士学位，导师就是 Knuth，现供职于贝尔实验室。

法类似，METAPOST 的主要优点在于是它输出的是 EPS，而且支持彩色；METAFONT 输出的是点阵格式，不支持彩色。Knuth 声称自己画图时只用 METAPOST。

从 Hobby 主页上 METAPOST 的更新记录看，它的最后版本是 0.63，年份是 1994。目前 Taco Hoekwater<sup>4</sup>继续 METAPOST 的开发工作，最新版本是 1.005。

本文只对 METAPOST 作简单介绍，若想深入了解请参阅 Hobby 的《A User's Manual for MetaPost》<sup>[2]</sup>。

#### 4.4.1 准备工作

用户一般需把 METAPOST 源文件（.mp）用一个命令行程序 `mpost` 编译为一种特殊的 EPS，也称作 MPS，然后再把 MPS 插入 L<sup>A</sup>T<sub>E</sub>X 源文件中使用。

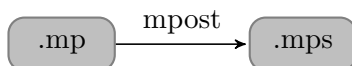


图 4.7: MetaPost 的编译

一个 METAPOST 源文件可以包含多个图形，一般形式如下。代码中每行语句以 ; 结尾，注释行以 % 起始。每个图形的绘图命令包含在一对起始和结尾声明之间。文件结尾也要有一个结尾声明。

```
beginfig(1); %图形起始
...          %绘图命令
endfig;      %图形结尾

beginfig(2);
...
endfig;
...
end;         %文件结尾
```

假如上面的源文件名字是 `fig.mp`，我们可以执行以下编译命令。

---

<sup>4</sup>他也是 LuaTeX 开发者之一。

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

编译后就会生成“fig.1、fig.2、...”等文件，每个文件的后缀就是相应的图形起始声明的编号。所以此编号在一个源文件中应保持唯一，否则后生成的文件就会覆盖前面的。

这样的文件名管理起来很麻烦，插入它们时也不能省略后缀，因为 $\text{\LaTeX}$ 不能识别它们。用`\DeclareGraphicsExtensions`来逐一声明后缀看起来很傻，自己改文件名更傻，`\DeclareGraphicsRule`也显得不够严谨。

然而`METAPOST`已经考虑到这个问题，为此提供了一个文件名模板命令。把下面的代码加到源文件头部，编译输出的文件名就会是“fig-01.mps、fig-02.mps、...”。

```
filenametemplate "%j-%2c.mps"; %加在源文件头部
```

我们也可以把这个命令加在每个图形的起始声明之前，指定个性化的输出文件名，这样可能更便于记忆。

```
filenametemplate "flowchart.mps" %加在每个图形前面
```

MPS 可以用`GSview`查看，我们也可以用以下命令把它转为PDF再用`Adobe Reader`查看。

```
epstopdf flowchart.mps
```

#### 4.4.2 基本图形对象

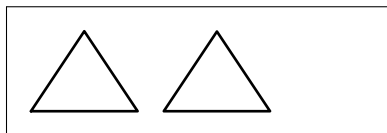
为了节省空间，本节后面的示例会略去图形起始声明和结尾声明等不重要的细节。

##### 直线

绘图命令`draw`把几个点以直线段连接起来。`METAPOST`中的缺省长度单位是`bp`，用户也可以使用表 2.5 中的其它单位。我们还可以定义一个缩放系数，把坐标都转换成此系数的倍数，这样以后想缩放图形时只要改这个系数即可。

注意 METAPOST 中的变量赋值符号是 `:=`，而 `=` 用于方程式。变量在同一源文件中只须定义一次，其后的图形中都可以使用。

```
draw (0,0)--(40,0)--(20,20)--(0,0);
u:=10pt; %缩放系数
draw (5u,0)--(9u,0)--(7u,2u)--cycle;
```

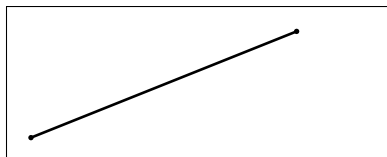


几段直线或曲线可以构成一条路径（path），在路径末尾加个 `cycle` 就能构成封闭路径（closed path）。上例中的两个三角形看起来都是封闭的，但是前面这个其实不是真正的封闭路径。

#### 4.4.3 点和线宽

`drawdot` 命令可以在指定坐标画一个点，为了使它醒目些我们可以换支粗一点的画笔。METAPOST 中的画笔缺省是直径 0.5pt 的圆形，拿它画出来的线宽就是 0.5pt。

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



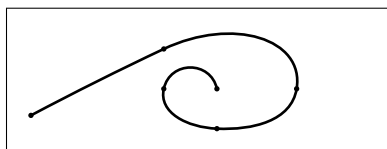
上面的 `pickup` 是一种全局操作，也就是说会影响到之后所有的绘图命令，我们也可以用 `withpen` 为单个绘图命令设置画笔。

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

#### 曲线

曲线和直线的命令相近，只是把连接两个点的 `--` 换成了 `..`。如果共用一些坐标，直线和曲线也可以混在一条语句里画。

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



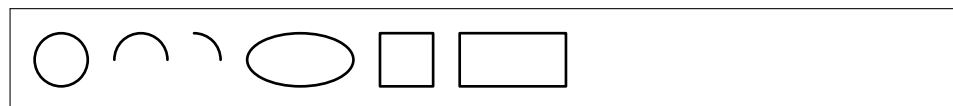
METAPOST 的曲线用三次贝塞尔 (Cubic Bézier) 算法实现。用户可以在命令中增加 `direction` (方向)、`Tension` (张力) 和 `Curl` (曲率) 等控制, 限于篇幅本文不赘述。

### 预定义图形

`fullcircle` 命令以原点为圆心画一个单位圆, 类似的预定义图形还有 `halfcircle`、`quartercircle`、`unitsquare` 等。注意单位正方形的参考点在左下而不在其中心。

通过不同的横向和纵向缩放系数, 我们可以把圆形和正方形变成椭圆和长方形。

```
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);
```

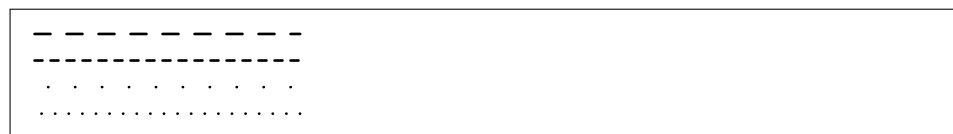


#### 4.4.4 图形控制

##### 线型和箭头

在绘制图形时, 我们不仅可以变换线宽, 也可以使用多种线型。

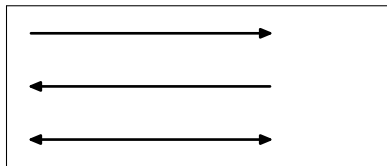
```
draw (0,0)--(10u,0) dashed withdots;
draw (0,1u)--(10u,1u) dashed withdots scaled 2;
draw (0,2u)--(10u,2u) dashed evenly;
draw (0,3u)--(10u,3u) dashed evenly scaled 2;
```





箭头和直线、曲线的语法相近，注意画反向箭头时需要把两个坐标用一对 `()` 括起来。

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

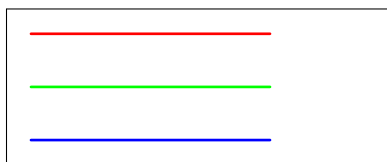


### 颜色和填充

METAPOST 预定义的颜色有黑、白、红、绿、蓝，它们的 RGB 值分别为(0,0,0)、(1,1,1)、(1,0,0)、(0,1,0)、(0,0,1)，缺省色就是黑色。

绘图命令一般都可以通过 `withcolor` 参数来使用各种颜色。封闭路径可以用 `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;
```



另一个命令 `filldraw` 可以看作是 `fill+draw`，它除了填充外还会把路径用指定的画笔画一遍。然而不幸的是画边缘和填充内部只能用同一种颜色，所以它的用处不大。

除了为每个绘图命令单独指定颜色，我们也可以使用一个全局命令，使得其后的绘图命令都使用某种颜色。

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

下面的方法可以用来定义基本色以外的颜色，用基本色混色或直接  
用RGB值效果是一样的。

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

### 图形变换

我们可以对路径进行缩放、平移、旋转等变换操作，横向和纵向缩放  
可以分开进行。由于旋转是围绕原点进行的，所以要注意平移和旋转的顺  
序。下例中定义了一个 `path` 变量，以便后面重用。

```
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);
```

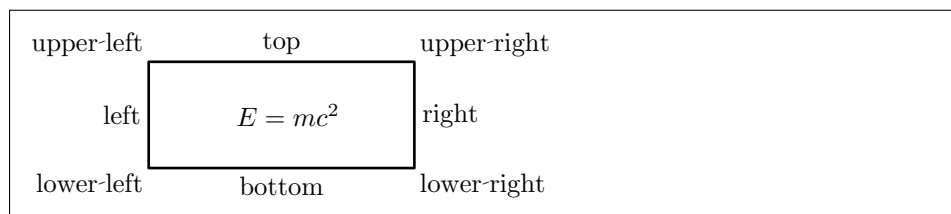


### 标注

`\label` 命令可以在指定的点附近加文字标注。METAPOST 也可以用一  
对 `btex` 和 `etex` 来嵌入一些  $\text{T}_\text{E}\text{X}$  内容，比如数学标注。

```
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));
```



因为用缺省方法编译生成的 MPS 不嵌入字体，当 METAPOST 包含文字时，GSview 就不能正常查看。这时我们可以给编译命令加个参数，生成的 MPS 就会包含字体信息。注意这种方法生成的 MPS 虽然 GSview 能查看，dvi2pdf 却不能正常处理。

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

#### 4.4.5 编程功能

##### 数据类型和变量

METAPOST 中有 10 种基本数据类型：numeric、pair、path、pen、color、cmykcolor、transform、string、boolean、picture。我们已经接触过其中几种，比如缩放系数 u 就是一个 numeric，一个点的坐标是一个 pair，几个点用直线或曲线连起来是一个 path，pencircle 是一种 pen，black 是一种 color，scaled、rotated、shifted 都是 transform。

numeric 类型变量的精度是 1/65536，它的绝对值不能超过 4096，在计算过程中数值可以达到 32768。这样的规定也应归功于当年的电脑硬件，不过对于科技文档插图而言，4096 一般还是够用的。

除了缺省的 numeric，其它变量在使用之前都需要用数据类型来显式声明。相同类型的变量可以在一行语句中声明，但是带下标的变量不能放在同一行（这个规定很蹊跷）。

```
numeric x,y,z;    %正确
numeric x1,x2,x3; %错误
numeric x[];      %正确
```

数学运算

METAPOST 中可以使用普通的运算符，比如 + - \* /；也提供一些特殊的运算符，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 列出一些常用数学函数。

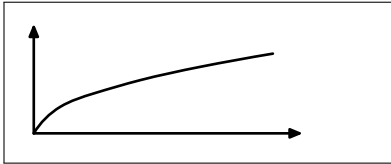
表 4.3: 数学函数

abs	绝对值	mexp	指数
round	四舍五入	mlog	对数
ceiling	向上圆整	sind	正弦
floor	向下圆整	cosd	余弦
mod	模余	normaldeviate	正态分布随机数
sqrt	开方	uniformdeviate	均匀分布随机数

循环

当执行重复任务时，循环语句可以让程序变得简洁。注意下例中的循环语句是一条命令，之所以分成三行写是为了看起来清晰点。

```
draw (0,0) %注意这里没有分号
for x=1 upto 3:
  ..(x*x,x)*u
endfor;
```



循环语句缺省步长是 1，我们也可以改用其它步长。upto 其实就是 step 1 until 的缩写方式。

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

## 4.5 PSTricks

PSTricks 是一个基于 PS 的宏包，有了它用户就可以直接在  $\text{\LaTeX}$  文档中插入绘图命令。PSTricks 早期的作者是 Timothy Van Zandt<sup>5</sup>，初始开发年月不详，他于 1997 年退居二线，之后由 Denis Girou、Sebastian Rahtz<sup>6</sup>、Herbert Voß 等维护。

本文只介绍 PSTricks 的基本功能，若想深入了解请参阅 Van Zandt 的《PSTricks User's Guide》<sup>[3]</sup>。另外表 4.4 列出了一些可以和 PsTricks 配合使用的辅助宏包。

表 4.4: PSTricks 辅助宏包

multido	循环语句	pst-eucl	几何函数
pst-plot	函数绘图	pst-math	弧度三角函数
pst-plot3d	三维绘图	pstricks-add	极坐标

### 4.5.1 准备工作

首先要引入 PSTricks 宏包。PSTricks 中缺省长度单位是 1cm，我们也可以设置自己的单位。

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

绘图命令一般要放在 `pspicture` 环境里，这样  $\text{\LaTeX}$  就会给图形预留一个矩形区域，注意这个矩形要能容纳所有图形对象。为了节省空间，在本节后面的示例代码中，`pspicture` 环境将被略去。

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

<sup>5</sup>法国Insead大学经济系教授。

<sup>6</sup>牛津大学计算机系网管。

另外需要注意的是，嵌入  $\text{\LaTeX}$  的 PSTricks 生成的是 PS，`dvips` 可以处理，`dvipdfm` 和 `pdf $\text{\LaTeX}$`  则不能。所以如果使用后两种 driver，需要先生成 EPS。

第一种方法是把 PSTricks 代码放进一个空白的  $\text{\LaTeX}$  页面，用它生成一个简单的 DVI。

```
\documentclass{article}
\usepackage{pstricks}
\pagestyle{empty} %页面样式为空

\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}
```

然后用以下命令把 DVI 转为 EPS，`-E` 参数即代表 EPS。上面代码中的 `colorbox` 使得生成的 EPS 有正确的范围框。

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

第二种方法是用 `pst-eps` 宏包，它能够在线处理 PSTricks 代码并生成 EPS，这样用户就可以在同一文件中使用该 EPS。

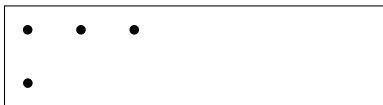
然而 `dvipdfmx` 不能正确处理 `pst-eps` 生成的 EPS。它和 `\rput`、`\uput` 命令，`pst-plot` 宏包中的 `\psaxes` 命令都不兼容。类似的 `ps4pdf` 宏包和 `tabularx` 宏包不兼容。

### 4.5.2 基本图形对象

#### 点

我们可以用以下命令画一个或多个点。

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



### 直线、多边形、矩形

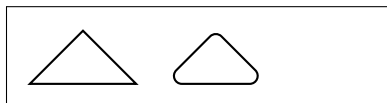
`\psline` 命令把多个点用直线段连接起来，线段之间的连接缺省为尖角，也可以设置圆角。

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



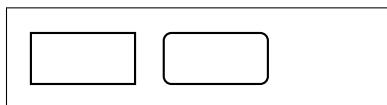
`\pspolygon` 命令和 `\psline` 类似，但是它会形成封闭路径。

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



矩形用 `\psframe` 命令，其参数就是矩形左下角和右上角的坐标。矩形也可以设置圆角。

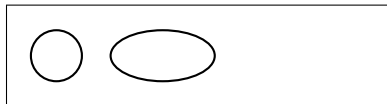
```
\psframe(0,0)(4,2)
\psframe[framearc=.3](5,0)(9,2)
```



### 圆、椭圆、圆弧、扇形

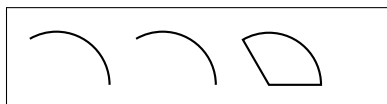
圆形用 `\pscircle` 命令，参数是圆心和半径。椭圆用 `\psellipse` 命令，参数是中心、长径、短径。注意这两个命令的半径参数用不同的括号，可能是作者的笔误。

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



圆弧用 `\psarc` 命令，其参数是圆心、半径、起止角度，逆时针作图。`\psarcn` 类似，只是顺时针作图。扇形用 `\pswedge` 命令。

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

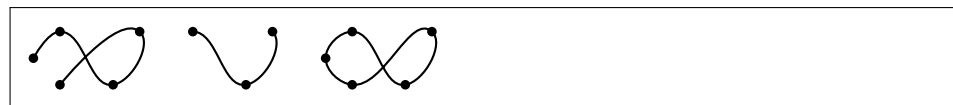


## 曲线

`\pscurve` 命令把一系列点用平滑曲线连接起来；它的变形版本 `\psecurve` 命令不显示曲线的两个端点；另一变形命令 `\psccurve` 则把曲线封闭起来。

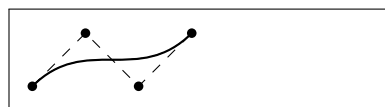
参数 `showpoints=true` 用来显示曲线的构成点，此参数也可用于其它绘图命令。

```
\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` 命令输出一条贝塞尔曲线，其参数就是曲线的控制点。

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



抛物线用 `\psparabola` 命令，它有两个参数，第一个是抛物线通过的一点，第二个是抛物线的顶点。

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

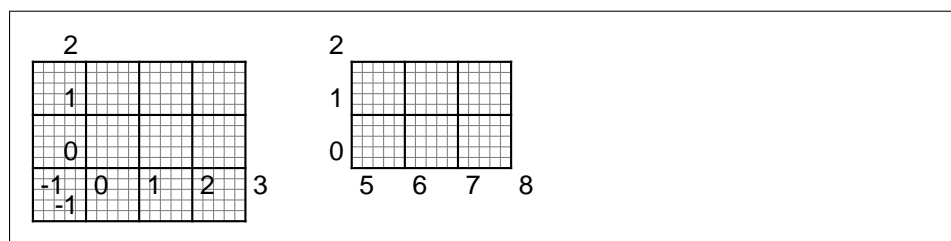


## 网格和坐标

科技制图通常会用到坐标网格。`\psgrid` 命令输出一个矩形网格，它有三个参数点。网格坐标标注在通过第一个点的两条直线上，第二和第三个点是矩形的两个对角顶点。当第一个参数省略时，坐标标注在通过第一个顶点的两条矩形边上。

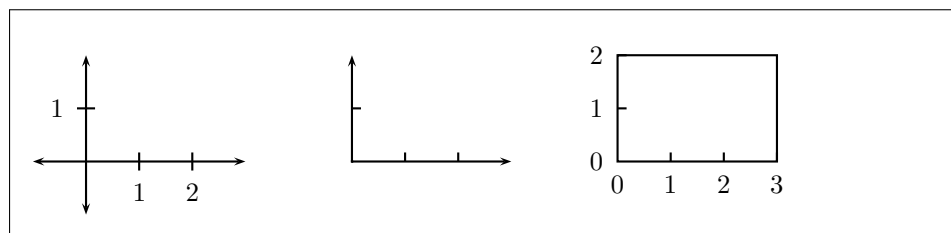
```
\psgrid(0,0)(-1,-1)(3,2)
\psgrid(5,0)(8,2)
```





`pst-plot` 宏包提供的 `\psaxes` 命令输出坐标轴。它的参数和 `\psgrid` 的类似，刻度和标注都可以灵活地设置，也可以把坐标轴改成一个矩形框的形式。

```
\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)
```

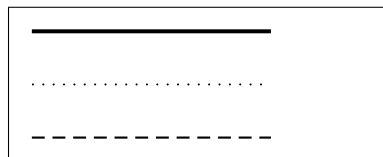


### 4.5.3 图形控制

#### 线型和箭头

PSTricks 中的缺省线宽是 0.8pt，缺省线型是实线。以下参数可以控制单个绘图命令的线宽和线型。

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

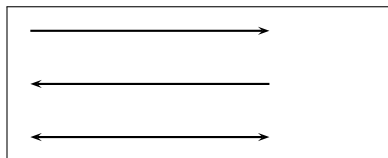


我们也可以用命令 `\psset` 命令来设置全局参数。

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

以下参数可以控制绘图命令的箭头。

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



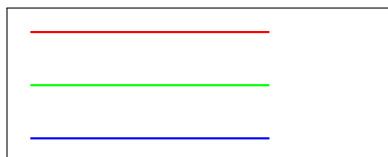
### 颜色和填充

PSTricks 预定义的颜色有 black、darkgray、gray、lightgray、white 等灰度颜色，也有 red、green、blue、cyan、magenta、yellow 等彩色。我们也可以自定义灰度颜色和彩色。

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

以下参数可以控制单个绘图命令的颜色，我们也可以用 \psset 命令设置全局参数。

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



以下参数可以控制单个绘图命令的填充模式和填充颜色，注意只有封闭路径才可以填充。

```
\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}
```

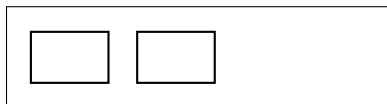


### 4.5.4 对象布局

#### 平移

参数 `origin` 可以让一个图形对象平移到指定的坐标点，我们也可以用 `\psset` 命令设置全局平移参数。

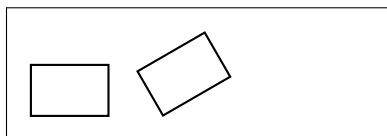
```
\psframe(0,0)(3,2)
\psframe[origin={4,0}](0,0)(3,2)
```



#### 旋转

`\rput` 命令可以对一个图形对象同时进行旋转和平移操作。它有两个参数，第一个是旋转角度，第二个是平移到的坐标点。

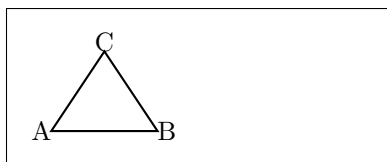
```
\psframe(0,0)(3,2)
\rput{30}(5,0){\psframe(0,0)(3,2)}
```



#### 文字标注

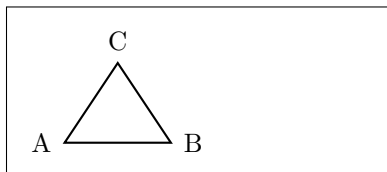
`\rput` 命令还可以在指定的坐标点标注文字，这时它的第一个参数是坐标点相对于标注的位置。其取值可以是纵向的 `t`、`b`（上下），或横向的 `l`、`r`（左右），也可以是纵向和横向位置的组合。

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



`\rput` 生成的标注就在坐标点上，有时会感觉离图形太近。另一个命令 `\uput` 则生成缺省距离指定坐标点 5pt 的标注，它的第一个参数是标注相对于坐标点的角度。

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



`\uput` 的角度参数可以是任意角度，也可以是字母，参见。注意 `\uput` 的角度参数和 `\rput` 命令的参考点位置参数的定义几乎正好相反，这也许反映了作者洒脱的风格。

表 4.5: `uput` 命令的角度参数

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<sup>7</sup>。Tantau 当初开发 Beamer 是为了应付 2003 年他的博士学位论文答辩，之后它在 CTAN 上流行开来。2005 年 PGF 从 Beamer 项目中分离出来，成为一个独立的宏包。

本文只对 PGF 作简单介绍，若想深入了解请参阅 Tantau 的《TikZ and PGF Manual》<sup>[4]</sup>。

### 4.6.1 准备工作

一般人们并不直接使用 PGF 命令，而是通过它前端 TikZ 来调用 PGF。在引用 `tikz` 宏包之前，用户需要设置 PGF 系统 driver。

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

PGF 的缺省长度单位是 1cm，我们也可以改用其它单位。注意这样预定义的长度单位有时会失效，这可能是 PGF 的 bug。

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

<sup>7</sup>德国 Lübeck 大学计算机研究所教授。

TikZ 提供一个 `tikz` 命令和一个 `tikzpicture` 环境，具体绘图指令可以放在 `tikz` 后面，也可以放在 `tikzpicture` 中间。两种方法效果相同，用户可以任意选择。为了节省空间，本节的示例将省略部分环境代码。

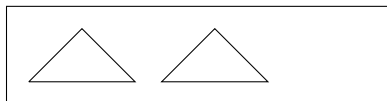
```
\tikz ... %绘图命令
\begin{tikzpicture}
... %绘图命令
\end{tikzpicture}
```

### 4.6.2 基本图形对象

#### 直线、多边形、矩形

TikZ 中直线的语法和 METAPOST 类似，加 `cycle` 参数才能构成真正的封闭路径。

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



矩形命令如下，它的两个参数是矩形的两个对角顶点。

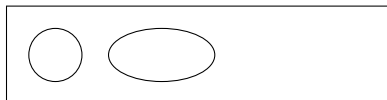
```
\draw (0,0) rectangle (4,2);
```



#### 圆、椭圆、弧

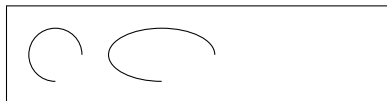
圆和椭圆命令如下，圆的参数是圆心和半径，椭圆的参数是中心、长径、短径。

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



圆弧和椭圆弧命令如下，圆弧的参数是起始点，起始角度、终止角度、半径；椭圆弧则把半径换成了长径和短径。

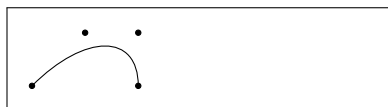
```
\draw (2,1) arc (0:270:1);
\draw (7,1) arc (0:270:2 and 1);
```



### 曲线和抛物线

曲线命令如下，中间参数是控制点。

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



抛物线命令如下，除了起止点还可以指定顶点。

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

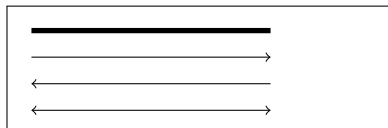


### 4.6.3 图形控制

#### 线型和箭头

绘图命令可以设置线型和箭头参数。

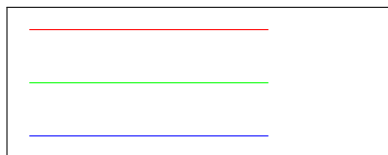
```
\draw[line width=2pt] (0,0)--(9,0);
\draw[->] (0,1)--(9,1);
\draw[<-] (0,2)--(9,2);
\draw[<->] (0,3)--(9,3);
```



#### 颜色、填充、阴影

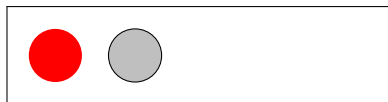
颜色参数的用法如下。PGF 可以使用 `xcolor` 宏包<sup>[5]</sup>中定义的所有颜色。

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



封闭路径可以用颜色填充，`\filldraw` 命令可以分别指定边框色和填充色。

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



`\shade` 命令可以产生渐变和光影效果，缺省是从上到下，灰色渐变为白色。我们也可以使用其它方向和颜色的渐变。

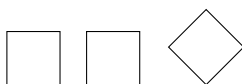
```
\shade (0,0) rectangle (2,2);
\shade[left color=red,right color=orange] (3,0) rectangle (5,2);
\shade[inner color=red,outer color=orange] (6,0) rectangle (8,2);
\shade[ball color=blue] (10,1) circle (1);
```



### 图形变换

对图形对象可以进行平移和旋转操作，注意如果两种操作同时进行，它们是有顺序的。注意预定义的长度单位在这里对平移参数失效。

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



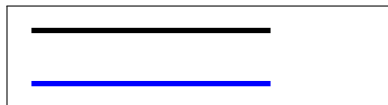
#### 4.6.4 样式

PGF 比 METAPOST 和 PSTricks 多了一个有趣的概念：样式（style），它的思路和 HTML 的 CSS 相近。我们可以先定义两种样式，

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

然后就可以在绘图命令中这样使用样式。

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



除了用 `\tikzset` 命令定义样式，我们也可以在 `tikzpicture` 环境头部声明样式。前者是全局性的，后者则是局部性的。

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

注意在样式中预定义长度单位有时会失效，所以最好使用绝对单位。

#### 4.6.5 流程图

##### 节点

PGF 中的节点 (node) 可以是简单的标签，也可以有各种形状的边框，还可以有各种复杂的属性。比如下例中的节点样式： `box`，它的边框是矩形，有圆角；它有最小宽度、高度、文字和边框的距离，边框和填充颜色等属性。

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

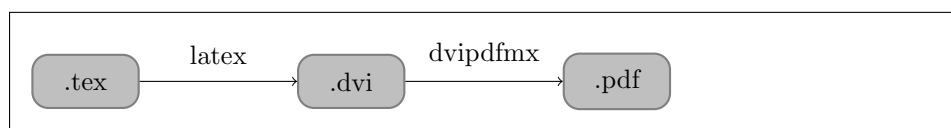
除了上述类别属性，节点还可以有名字、位置等属性。在下例中，我们先画了三个有名字的文本框；然后用箭头把文本框连接起来，注意连接时要引用文本框的名字；接着在箭头上加了标签。



```

\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);          %箭头
\draw[->] (dvi)--(pdf);          %箭头
\node at (5,1) {latex};          %标签
\node at (15,1) {dvi pdfmx};     %标签

```



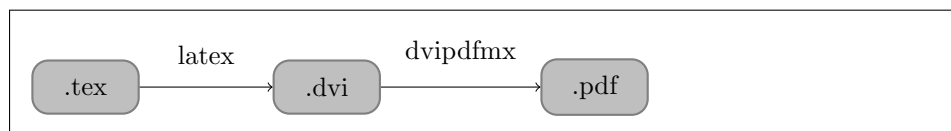
在上例中的节点都使用了绝对位置，PGF 中还可以使用更灵活一点的相对位置。比如在下例中，dvi 节点在 tex 节点右边 50pt 处（我们前面定义的基本长度单位是 10pt），而 pdf 节点又在 dvi 节点右边 50pt 处。

箭头可以换为专门用来连接节点的 `edge`；标签也改成相对位置，箭头上方 5pt 处。

```

\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] {dvi pdfmx} (pdf);

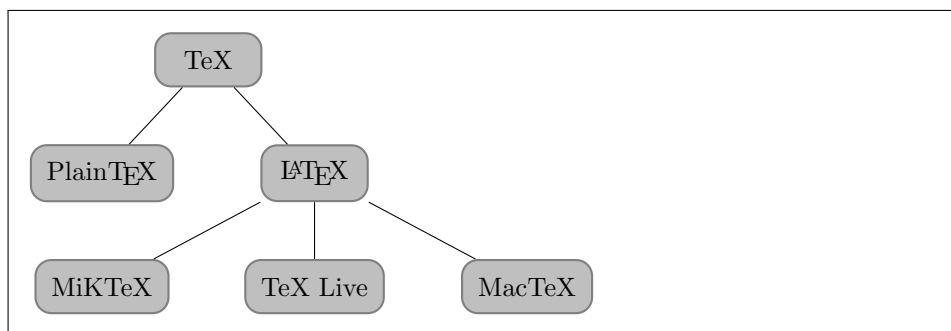
```



## 树

下面是一棵简单的树。我们可以用一个参数控制相邻节点的距离，预定义长度单位对此参数也会失效。

```
\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}
```



## 参考文献

- [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 简单表格

`tabular` 环境提供了最简单的表格功能。它用 `\hline` 命令代表横线，`|` 代表竖线，用 `&` 来分栏。每个栏位的对齐方式可以用 `l`、`c`、`r`（左中右）来控制。

```
\begin{tabular}{|l|c|r|}  
  \hline  
  操作系统 & 发行版 & 编辑器 \\  
  \hline  
  Windows & MikTeX & TeXnicCenter \\  
  \hline  
  Unix/Linux & TeX Live & Emacs \\  
  \hline  
  Mac OS & MacTeX & TeXShop \\  
  \hline  
\end{tabular}
```

操作系统	发行版	编辑器
Windows	MikTeX	TeXnicCenter
Unix/Linux	TeX Live	Emacs
Mac OS	MacTeX	TeXShop

和针对插图的 `figure` 环境类似， $\text{\LaTeX}$  还有另一个针对表格的浮动环境 `table`。我们可以用它给上面的示例穿件马甲，顺便把表格简化为科技文献中常用的三线表。

```

\begin{table}[htbp]
\caption{浮动环境中的三线表}
\label{tab:threesome}
\centering
\begin{tabular}{lll}
\hline
操作系统 & 发行版 & 编辑器 \\
\hline
Windows & MikTeX & TeXnicCenter \\
Unix/Linux & TeX Live & Emacs \\
Mac OS & MacTeX & TeXShop \\
\hline
\end{tabular}
\end{table}

```

表 5.1: 浮动环境中的三线表

操作系统	发行版	编辑器
Windows	MikTeX	TeXnicCenter
Unix/Linux	TeX Live	Emacs
Mac OS	MacTeX	TeXShop

完美主义者可能觉得上面示例中的三条线一样粗不够美观，这时可以使用 `booktabs` 宏包<sup>[1]</sup>的几个命令。

```

\begin{table}[htbp]
\caption{浮动环境中的三线表}
\centering
\begin{tabular}{lll}
\toprule
操作系统 & 发行版 & 编辑器 \\
\midrule
Windows & MikTeX & TeXnicCenter \\
\midrule
Unix/Linux & TeX Live & Emacs
\end{tabular}
\end{table}

```

```

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

```

表 5.2: booktabs 宏包的效果

操作系统	发行版	编辑器
Windows	MikTeX	TeXnicCenter
Unix/Linux	TeX Live	Emacs
Mac OS	MacTeX	TeXShop

## 5.2 表格宽度

有时我们需要控制某栏位宽度，可以将其对齐方式参数从 `l`、`c`、`r` 改为 `p{宽度}`。

```

\begin{table}[htbp]
\caption{控制栏位宽度}
\centering
\begin{tabular}{p{100pt}p{100pt}p{100pt}}
\toprule
操作系统 & 发行版 & 编辑器 \\
\midrule
Windows & MikTeX & TeXnicCenter \\
Unix/Linux & TeX Live & Emacs \\
Mac OS & MacTeX & TeXShop \\
\bottomrule
\end{tabular}
\end{table}

```

若想控制整个表格的宽度可以使用 `tabularx` 宏包，`x` 参数表示某栏可以折行。

表 5.3: 控制栏位宽度

操作系统	发行版	编辑器
Windows	MikTeX	TeXnicCenter
Unix/Linux	TeX Live	Emacs
Mac OS	MacTeX	TeXShop

```

\begin{table}[htbp]
\caption{控制表格宽度}
\centering
\begin{tabularx}{350pt}{lXlX}
\toprule
李白 & 平林漠漠烟如织，寒山一带伤心碧。暝色入高楼，有人楼上愁。
玉梯空伫立，宿鸟归飞急。何处是归程，长亭更短亭。&
泰戈尔 & 夏天的飞鸟，飞到我的窗前唱歌，又飞去了。秋天的黄叶，它们
没有什么可唱，只叹息一声，飞落在那里。\\
\bottomrule
\end{tabularx}
\end{table}

```

表 5.4: 控制表格宽度

李白	平林漠漠烟如织，寒山一带伤心碧。暝色入高楼，有人楼上愁。玉阶空伫立，宿鸟归飞急。何处是归程，长亭更短亭。	泰戈尔	夏天的飞鸟，飞到我的窗前唱歌，又飞去了。秋天的黄叶，它们没有什么可唱，只叹息一声，飞落在那里。
----	--	-----	---

### 5.3 跨行、跨列表格

有时某栏需要横跨几列，我们可以使用 `\multicolumn` 命令。它的前两个参数指定横跨列数和对齐方式。`booktabs` 宏包的 `\cmidrule` 命令用于横

跨几列的横线。

```
\begin{table}[htbp]
\caption{跨栏表格}
\centering
\begin{tabular}{lll}
\toprule
& \multicolumn{2}{c}{常用工具} \\
\cmidrule{2-3}
操作系统 & 发行版 & 编辑器 \\
\midrule
Windows & MikTeX & TeXnicCenter \\
Unix/Linux & TeX Live & Emacs \\
Mac OS & MacTeX & TeXShop \\
\bottomrule
\end{tabular}
\end{table}
```

表 5.5: 跨栏表格

操作系统	常用工具	
	发行版	编辑器
Windows	MikTeX	TeXnicCenter
Unix/Linux	TeX Live	Emacs
Mac OS	MacTeX	TeXShop

跨行表格需要使用 `multirow` 宏包, `\multirow` 命令的前两个参数是竖跨的行数和宽度。

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

```

\toprule
操作系统 & 发行版 & 编辑器 & 用户体验\\
\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

操作系统	发行版	编辑器	用户体验
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<sup>[2]</sup>提供的一些命令：`\columncolor`、`\rowcolor`、`\cellcolor` 等。

```

\usepackage{colortbl}
...
\begin{table}[htbp]
\caption{彩色表格}
\centering
\begin{tabular}{llll}
\toprule
操作系统 & 发行版 & 编辑器 & \\
\midrule
Windows & MikTeX & TeXnicCenter & \\
\end{tabular}

```



```

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

```

表 5.7: 彩色表格

操作系统	发行版	编辑器
Windows	MikTeX	TeXnicCenter
Unix/Linux	TeX Live	Emacs
Mac OS	MacTeX	TeXShop

## 5.5 长表格

有时表格太长要跨页，可以使用 `longtable` 宏包<sup>[3]</sup>。`\endfirsthead`、`\endhead` 命令用来定义首页表头和通用表头，`\endfoot`、`\endlastfoot` 命令用来定义通用表尾和末页表尾。

```

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

```

```

\endfoot
\bottomrule
\endlastfoot
白居易 & 汉皇重色思倾国，\\
& 御宇多年求不得。\\
& 杨家有女初长成，\\
& 养在深闺人未识。\\
& 天生丽质难自弃，\\
& 一朝选在君王侧。\\
& 回眸一笑百媚生，\\
& 六宫粉黛无颜色。\\
& 春寒赐浴华清池，\\
& 温泉水滑洗凝脂。\\
& 侍儿扶起娇无力，\\
& 始是新承恩泽时。\\
& 云鬓花颜金步摇，\\
& 芙蓉帐暖度春宵。\\
& 春宵苦短日高起，\\
& 从此君王不早朝。\\
\end{longtable}

```

表 5.8: 长表格

作者	作品
白居易	汉皇重色思倾国， 御宇多年求不得。 杨家有女初长成， 养在深闺人未识。 天生丽质难自弃， 一朝选在君王侧。 回眸一笑百媚生， 六宫粉黛无颜色。

接下页...

---

作者	作品
	春寒赐浴华清池， 温泉水滑洗凝脂。 侍儿扶起娇无力， 始是新承恩泽时。 云鬓花颜金步摇， 芙蓉帐暖度春宵。 春宵苦短日高起， 从此君王不早朝。

---

## 参考文献

- [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>.

看什么看，没见过空白页？

## 第六章 杂项

### 6.1 超链接

`hyperref` 宏包<sup>[1]</sup>提供了一些超链接功能。它给文档内部的交叉引用和参考文献自动加上了超链接，还提供了几个命令。

`\hyperref` 命令对已经定义的label进行简单包装，加上文字描述。

```
\usepackage{hyperref}
...
\label{sec:hyperlink}
...
```

例如`\ref{sec:hyperlink}`是编号形式的链接，而`\hyperref{sec:hyperlink}{这个链接}`是文字形式的链接，都指向本节开始。

例如6.1是编号形式的超链接，而[这个链接](#)则是文字形式，都指向本节开始。

`\url` 和 `\href` 命令可以用来定义外部链接，后者有文字描述。

```
\url{http://www.dralpha.com/}
\href{http://www.dralpha.com/}{包老师的主页}
```

<http://www.dralpha.com/>  
包老师的主页

## 6.2 长文档

当文档很长时，我们可以把它分为多个文件，然后在主控文档的正文中引用它们。注意 `\include` 命令会新起一页，如果不想要新页可以改用 `\input` 命令。

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

当文档很长时，编译一遍也会很花时间，我们可以用 `syntonly` 宏包。这样编译时就只检查语法，而不生成结果文件。

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

## 6.3 参考文献

在文档中，我们经常要引用参考文献（bibliography）。 $\text{\LaTeX}$  提供的 `thebibliography` 环境和 `\bibitem` 命令可以用来定义参考文献条目及其列表显示格式，`cite` 命令用来在正文中引用参考文献条目。这种方法把内容和格式混在一起，用户需要为每个条目设置格式，很繁琐且易出错。

### 6.3.1 BibTeX

1985年，Oren Patashnik<sup>1</sup>和 Lamport 开发了  $\text{BibTeX}$ <sup>[2]</sup>，其详细使用方法请参阅 Nicolas Markey 的《Tame the BeaST: The B to X of BibTeX》<sup>[3]</sup>

---

<sup>1</sup>Wiki 上说他是 Knuth 的学生，我发现他不在 Knuth 的博士生列表上，而在姚期智的博士生列表上，也许他是 Knuth 的硕士生。

BibTeX 把参考文献的数据放在一个 .bib 文件中，显示格式放在 .bst 文件中。普通用户一般不需要改动 .bst，只须维护 .bib 数据库。

一个 .bib 文件可以包含多个参考文献条目 (entry)，每个条目有类型、关键字，以及题目、作者、年份等字段。常用条目类型有 article、book、conference、manual、misc、techreport 等。每种类型都有一些自己的规定字段和可选字段，字段之间用逗号分开。数据库中每个条目的关键字要保持唯一，因为引用时要用到它们。

下例显示了一个条目，它的类型是 manual，关键字是 Markey\_2005。 .bib 文件可以用普通文本编辑器来编辑，也可以用专门的文献管理软件来提高效率。包老师推荐 [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/}
}
```

有了数据库，我们可以象下面这样引用一个条目。

请参阅\cite{Markey\_2005}。

请参阅[3]。

前文中我们提到含有交叉引用的文档需要编译两遍。含有参考文献的文档更麻烦，它需要依次执行 latex、bibtex、latex、latex 等四次编译操作。

1. 第一遍 latex 只把条目的关键字写到中间文件 .aux 中去。
2. bibtex 根据 .aux、.bib、.bst 生成一个 .bbl 文件，即参考文献列表。它的内容就是 thebibliography 环境和一些 \bibitem 命令。
3. 第二遍 latex 把交叉引用写到 .aux 中去。
4. 第三遍 latex 则在正文中正确地显示引用。

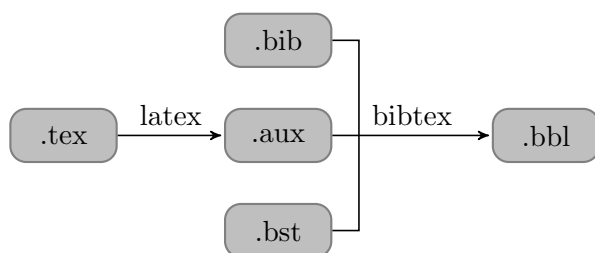


图 6.1: BibTeX 的编译

注意在长文档中使用参考文献时，应该用 `latex` 编译主控文档，而用 `bibtex` 编译子文档。

```

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

```

### 6.3.2 natbib

参考文献的引用通常有两种样式：作者-年份和数字。L<sup>A</sup>T<sub>E</sub>X 本身只支持数字样式，而 `natbib` 宏包<sup>[4]</sup>则同时支持这两种样式。

使用 `natbib` 宏包时，我们首先要引用宏包；其次设置文献列表样式和引用样式，每种列表样式都有自己的缺省引用样式，所以后者可选；然后指定参考文献数据库。

```

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

```

`natbib` 提供了三种列表样式：`plainnat`、`abbrvnat`、`unsrtnat`。前两种都是作者-年份样式，文献列表按作者-年份排序，后者会使用一些缩写（比



如作者的 first name)；unsrtnat 是数字样式，文献列表按引用顺序排序。

`\setcitestyle` 命令可以用来改变引用样式的设置，其选项见 表 6.1。

表 6.1: 参考文献引用样式选项

引用模式	authoryear、numbers、super
括号	round、square、open=char,close=char
引用条目分隔符	分号、逗号、citesep=char
作者年份分隔符	aysep=char
共同作者年份分隔符	yysep=char
注解分隔符	notesep=text

注意在长文档中，每个含参考文献的子文档都需要分别设置列表样式，并指定数据库。

`natbib` 提供了多种引用命令，其中最基本的是 `\citet` 和 `\citep`，它们在不同引用模式下效果不同。一般不推荐使用  $\text{\LaTeX}$  本身提供的 `\cite`，因为它在作者-年份模式下和 `\citet` 一样，在数字模式下和 `\citep` 一样。

作者-年份模式下引用命令的效果如下。

参阅 <code>\cite{Daly_2007}\</code>	参阅Daly [2007]
参阅 <code>\citet{Daly_2007}\</code>	参阅Daly [2007]
参阅 <code>\citep{Daly_2007}</code>	参阅[Daly, 2007]

数字模式下引用命令的效果如下。

参阅 <code>\cite{Daly_2007}\</code>	参阅[4]
参阅 <code>\citet{Daly_2007}\</code>	参阅Daly [4]
参阅 <code>\citep{Daly_2007}</code>	参阅[4]

上标模式下引用命令的效果如下。

参阅 <code>\cite{Daly_2007}\</code>	参阅 <sup>[4]</sup>
参阅 <code>\citet{Daly_2007}\</code>	参阅Daly <sup>[4]</sup>
参阅 <code>\citep{Daly_2007}</code>	参阅 <sup>[4]</sup>

另外还有一些引用命令，如 `\citetext`、`\citenum`、`\citeauthor`、`\citeyear` 等，此处不赘述。

## 6.4 索引

`makeidx` 宏包提供了索引功能。应用它时，我们首先需要在文档序言部分引用宏包，并使用 `makeindex` 命令；其次在正文中需要索引的地方定义索引，注意索引关键字在全文中须保持唯一；最后在合适的地方（一般是文档末尾）打印索引。

```
\usepackage{makeidx}
\makeindex
...
\begin{document}
\index{索引关键字}
...
\printindex
\end{document}
```

当编译含索引的文档时，用户需要执行 `latex`、`makeindex`、`latex` 等三次编译操作。

1. 第一遍 `latex` 把索引条目写到一个 `.idx` 文件中。
2. `makeindex` 把 `.idx` 排序后写到一个 `.ind` 文件中。
3. 第二遍 `latex` 在 `\printindex` 命令的地方引用 `.ind` 的内容，生成正确的DVI。

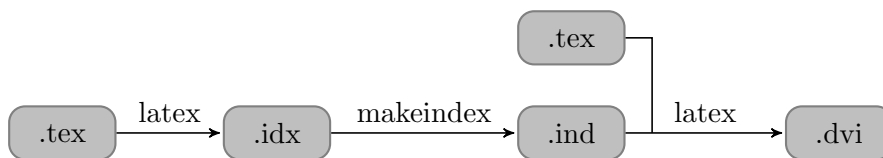


图 6.2: 索引的编译

6.5 页面布局

在 L<sup>A</sup>T<sub>E</sub>X 中用户可以通过 `\pagestyle` 和 `\pagenumbering` 命令来设置页眉 (header)、页脚 (footer) 的样式和内容。页面样式有以下四种。

表 6.2: L<sup>A</sup>T<sub>E</sub>X 页面样式

<code>empty</code>	页眉、页脚空白
<code>plain</code>	页眉空白, 页脚含居中页码
<code>headings</code>	页脚空白, 页眉含章节名和页码
<code>myheadings</code>	页脚空白, 页眉含页码和用户自定义信息

`fancyhdr`<sup>[5]</sup>宏包提供了更灵活的控制。我们可以用以下代码定制页眉、页脚的内容, 以及页眉下方、页脚上方的横线。

```
\usepackage{fancyhdr}
...
\pagestyle{fancy} %fancyhdr宏包新增的页面风格
\lhead{左擎苍}
\chead{三个代表}
\rhead{右牵黄}
\lfoot{左青龙}
\cfoot{八荣八耻}
\rfoot{右白虎}
\renewcommand{\headrulewidth}{0.4pt}
\renewcommand{\footrulewidth}{0.4pt}
```

左擎苍	三个代表	右牵黄
和谐社会		
左青龙	八荣八耻	右白虎

用户可以在页眉、页脚中使用一些  $\text{\LaTeX}$  变量，比如分别代表页码和章节编号的 `\thepage`、`\thechapter`、`\thesection`；代表章节起始单词（Chapter、Section等）的 `\chaptername`、`\sectionname` 等。

这些变量组合起来可以构成复合标记 `\leftmark` 和 `\rightmark`。当文档奇偶页面布局不同时，我们可以使用以下方法为奇偶页分别设置页眉、页脚。`fancyhdr` 宏包会自动把每章起始页的样式设为 `plain`，若想去掉页脚中间的页码，可以重定义 `plain` 样式。

```
\pagestyle{fancy}
\fancyhf{}           %清空页眉页脚
\fancyhead[LE,RO]{\thepage} %偶数页左，奇数页右
\fancyhead[RE]{\leftmark}  %偶数页右
\fancyhead[LO]{\rightmark} %奇数页左
\fancypagestyle{plain}{    %重定义plain页面样式
  \fancyhf{}
  \renewcommand{\headrulewidth}{0pt}
}
```

3.2 节名	17
奇数页	

18	Chapter 3 章名
偶数页	

Lamport当初设计  $\text{\LaTeX}$  时把页面布局变量的定义方式搞得比较晦涩，用户在重定义 `\leftmark` 和 `\rightmark` 时，不能直接用 `\renewcommand`

的方法，而要用另外两个命令。

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

`\leftmark` 即 `main-mark`，是一种高层次标记，在 `article` 文档类中它包含 `section` 的信息，在 `report` 和 `book` 则包含 `chapter` 的信息；`rightmark` 则是一种低层次标记，在 `article` 中包含 `subsection` 信息，在 `report` 和 `book` 中包含 `section` 信息。

比如在 `book` 文档类中，章节标记是通过下面的方法定义的，其中的 `#1` 指的是章节的名字。

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

## 参考文献

- [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/local1tx.html>.
- [5] Piet van Oostrum. *Page Layout in LaTeX*, 2004. URL <http://tug.ctan.org/tex-archive/macros/latex/contrib/fancyhdr/>.

看什么看，没见过空白页？

# 第七章 中文

## 7.1 字符集和编码

众所周知电脑内部采用二进制编码，因为它易于用电子电路实现。所有字符（包括字母、数字、符号、控制码等）在电脑内部都是用二进制表示的，字符集（Character Set）的二进制编码被称为字符编码（Character Encoding），有时人们也会混用这两个术语。

1963 年发布的 American Standard Code for Information Interchange（ASCII）是最早出现的字符编码，它用 7 位（bit）表示了  $2^7 = 128$  个字符，只能勉强覆盖英文字符。

美国人发明了电脑，英语被优先考虑是很自然的事情。随着电脑技术的传播，人们呼吁把字符编码扩充到 8 位也就是一个字节（byte），于是国际标准化组织（International Organization for Standardization, ISO）推出了 ISO 8859。 $2^8 = 256$  个字符显然也不能满足需要，所以 8859 被分为十几个部分，从 8859-1（西欧语言）、8859-2（中欧语言），直到 8859-16（东南欧语言），覆盖了大部分使用拉丁字母的语言文字。

在 ISO 标准完全定型之前，IBM 就有一系列自己的字符编码，他们称之为代码页（Code Page），其中著名的有 1981 年就被用于 IBM PC 的 437（扩展 ASCII）、850（西欧语言）、852（东欧语言）。IBM 代码页通常被用于控制台（Console）环境，也就是 MS-DOS 或 Unix Shell 那样的命令行环境。

微软将 IBM 代码页称为 OEM 代码页，自己定义的称为 ANSI 代码页，后者中著名的有 1252（西欧语言）、1250（东欧语言）、936（GBK

简体中文)、950 (Big5 繁体中文)、932 (SJIS 日文)、949 (EUC-KR 韩文) 等。

1981 年, 中国大陆推出了第一个自己的字符集标准 GB2312, 它是一个  $94 \times 94$  的表, 包括 7445 个字符 (含 6763 个汉字)。GB2312 通常采用双字节的 EUC-CN 编码, 所以后者也常常被称为 GB2312 编码; 其实 GB2312 还有另一种编码方式 HZ, 只是不常用。GB2312 不包含朱镕基的“镕”字, 政府、新闻、出版、印刷等行业和部门在使用中感到十分不便, 于是它在 1993 年被扩展为 GBK, 后者包括 21886 个字符 (含 21003 个汉字), 没有形成正式标准。2000 年发布的 GB18030 包含 70244 个字符 (含 27533 个汉字), 采用四字节编码。GB18030 之前还出现过一个 GB13000, 没有形成气候。

1990 年 ISO 推出了通用字符集 (Universal Character Set, UCS), 即 ISO 10646, 意图一统江湖。它有两种编码: 双字节的 UCS-2 和四字节的 UCS-4。

ISO 之外还有个希望一统江湖的组织: 统一码联盟 (The Unicode Consortium), 它于 1991 年推出了 Unicode 1.0。后来两家组织意识到没必要做重复工作, 于是开始合并双方的成果, 携手奔小康。从 Unicode 2.0 开始, Unicode 采用了与 ISO 10646-1 相同的编码。

Unicode 主要有三种编码: UTF-8、UTF-16、UTF-32。UTF-8 使用一至四个 8 位编码。互联网工程任务组 (Internet Engineering Task Force, IETF) 要求所有网络协议都支持 UTF-8, 互联网电子邮件联盟 (Internet Mail Consortium, IMC) 也建议所有电子邮件软件都支持 UTF-8, 所以它已成为互联网上的事实标准。UTF-16 用一或两个 16 位编码, 基本上是 UCS-2 的超集, 和 ASCII 不兼容。UTF-32 用一个 32 位编码, 它是 UCS-4 的一个子集。

## 7.2 中文解决方案

$\text{\TeX}$  是基于单字节编码的, 因为 Knuth 当初开发  $\text{\TeX}$  时没考虑那么远, 也没有现成的标准可以借鉴。

$\text{\LaTeX}$  对中文的支持主要有两种方法: 张林波<sup>1</sup>开发的 CCT 和 Werner

---

<sup>1</sup>中科院计算数学所某实验室主任。



Lemberg<sup>2</sup>开发的 CJK 宏包。早期 CCT 比较流行，新的 CCT 也可以和 CJK 配合使用，网上可以找到的最后更新是 2003 年的。CJK 是当前主流，它不仅支持中日韩等东亚文字，还支持几十种其他不同语言的多种编码。

支持简体中文的 L<sup>A</sup>T<sub>E</sub>X 发行版有吴凌云<sup>3</sup>的 CTeX 和李树钧<sup>4</sup>的 ChinaTeX，繁体中文的有吴聪敏<sup>5</sup>、吴聪慧兄弟的 cwTeX 和蔡奇伟<sup>6</sup>的 PUTeX。后面两个台湾的发行版包老师不熟悉，前面两个大陆的发行版都包含 MikTeX、CCT、CJK、WinEdt 等。

## 7.3 CJK的使用

CJK 的作者写的使用说明比较凌乱和晦涩，读者可以参阅李果正的《我的CJK》<sup>[1]</sup>。

CJK 有两个基本宏包：CJK 和 CJKutf8，后者面向 UTF-8 编码。CJK 环境的一般使用方法如下：

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

简体中文常用编码是 GBK 和 UTF8。family 是指宋体、楷体、隶书等，具体引用要看电脑上安装了什么字体。麻烦的是 GBK 和 UTF8 字体不通用，也就是说每种编码需要自己的字体。CJK 自带的 UTF8 简体字体有 gbsn（宋体）和 gkai（楷体）。CTeX 提供的 GBK 字体有 song（宋

<sup>2</sup>1968 年生于奥地利，从维也纳音乐学院获得作曲、指挥、钢琴、乐团管理、歌手教练等五个专业文凭，后自学中文和数学。曾任职于奥地利和德国多家剧院和乐团，现任德国科布伦茨某剧院指挥。

<sup>3</sup>中科院应用数学所研究员。

<sup>4</sup>德国哈根函授大学研究员。

<sup>5</sup>台湾大学电机系学士，美国罗彻斯特大学经济学博士，现任台湾大学经济系教授。

<sup>6</sup>美国犹他大学计算机博士，现任台湾静宜大学资讯工程系教授。

体)、fs(仿宋)、kai(楷体)、hei(黑体)、li(隶书)、you(幼圆)等。

注意使用 CJKutf8 宏包时, CJK 环境的编码最好是 UTF8, 否则可能遭遇不测。

下面是一个简单的 GBK 示例和一个 UTF8 示例, 注意用编辑器保存时要选则相应的编码格式, 比如前者用 ANSI, 后者用 UTF-8。

```
\documentclass{article}
\usepackage{CJK}
\begin{document}
\begin{CJK}{GBK}{song}
这是一个CJK例子, 使用了GBK编码和song字体。
\end{CJK}
\end{document}
```

```
\documentclass{article}
\usepackage{CJKutf8}
\begin{document}
\begin{CJK}{UTF8}{gbsn}
这是一个CJK例子, 使用了UTF-8编码和gbsn字体。
\end{CJK}
\end{document}
```

## 参考文献

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

# 第八章 字体

关于字体有三个重要概念：glyph、typeface、font。glyph 通常被翻译为字形，也有翻译为字体的；typeface 是一个书法和印刷领域的概念，它通常被翻译为字体或书体；font 曾经和 typeface 混用，但现在一般用作电脑领域的概念，在中国大陆被翻译为字体，在台湾被翻译为字型。

上述翻译的混乱令人十分无奈，包老师决定在本文中把它们分别翻译为字形、字样、字体，以正视听。

字形是一个字符的具体图形表现形式，一个字符可以有多个字形，比如汉字中的“強 / 强”、“戶 / 户 / 戸”；字样是一组相同风格样式的字形的集合，比如中文字样有宋、仿、楷、黑、隶、篆等；一种字样可以对应电脑上的几种字体。

## 8.1 字样

拉丁字母的字样主要有三大类：Serif（Roman）、Sans Serif 和 Monospace（Typewriter）。Serif 的笔画边缘部分有些装饰，类似于中文的宋体、仿宋、楷体、魏书等。Sans Serif 的笔画则是平滑的，类似于中文的黑体。Sans 这个词来源于法语，就是“没有”的意思。Monospace 则是等宽字样。

每一类字样都可以有加粗（bold）、斜体（italic）、倾斜（oblique）等修饰效果。Italic 通常对原字样进行过重新设计，它修饰精细，多用于 Serif；Oblique 也称作 slanted，基本上是把正体倾斜，多用于 Sans Serif。通常 oblique 看起来比 italic 要宽一些。

表 8.1 列出了几种常见的字样。

表 8.1: 常见字样

操作系统	Serif	Sans Serif	Monospace
Mac OS	Times	Helvetica	Courier
Windows	Times New Roman	Arial	Courier New

## 8.2 字体格式

### 8.2.1 点阵字体和矢量字体

电脑上用的字体（font）按数据格式可以分为三大类：点阵字体（bitmap）、轮廓（outline）字体和笔画（stroke-based）字体。

点阵字体通过点阵来描述字形。早期的电脑受到容量和绘图速度的限制，多采用点阵字体。点阵字体后来渐渐被轮廓字体所取代，但是很多小字号字体仍然使用它，因为这种情况下轮廓字体缩放太多会导致笔画不清晰。

轮廓字体又称作矢量字体，它通过一组直线段和曲线来描述字形。轮廓字体易于通过数学函数进行缩放等变换，形成平滑的轮廓。轮廓字体的主要缺陷在于它所采用的贝塞尔曲线（Bézier curves）在光栅（raster）设备（比如显示器和打印机）上不能精确渲染，因而需要额外的补偿处理比如字体微调（font hinting）。但是随着电脑硬件的发展，人们一般不在意它比点阵字体多出的处理时间。

笔画字体其实也是轮廓字体，不过它描述的不是完整的字形，而是笔画。它多用于东亚文字。

### 8.2.2 常见字体

常见的轮廓字体技术有：Type 1 和 Type 3、TrueType、OpenType、METAFONT 等。

Adobe 的 Type 1 和 Type 3 基于 PS，它们采用三次贝塞尔曲线。Type 1 支持微调，它使用一个简化的 PS 子集；Type 3 不支持微调，但它可以使用全部 PS 功能，因此既可以包含轮廓字体也可以包含点阵字体信息。

1991 年，Apple 发布了 TrueType，它采用二次贝塞尔曲线。二次曲线处理起来比三次曲线快，但是需要更多的点来描述。所以从 TrueType 到 Type 1 的转换是无损的，反之是有损的。1994 年，Apple 着手研究 TrueType 的下一代技术：TrueType GX，它后来演变为 Apple Advanced Typography (AAT)。

1996 年，微软和 Adobe 联合发布了 OpenType。它比起 AAT 的优势有：跨平台、开放和易于开发、支持更多的语言比如阿拉伯语。

早在 1984 年 Knuth 就发布了 METAFONT，它与 TrueType 和 OpenType 的区别是，不直接描述字形轮廓，而描述生成轮廓的笔的轨迹。笔的形状可以是椭圆形或多边形，尺寸缩放自如，字形边缘也柔和一些。两种字体可以用同一个 METAFONT 文件，当然还有不同的参数。METAFONT 技术如此先进，却没有流行开来。对此 Knuth 解释道，要求一位设计字体的艺术家掌握 60 个参数太变态了，那是用来折磨数学家的。

Type 1 和 Type 3 把字体信息存储在两种文件里：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)。L<sup>A</sup>T<sub>E</sub>X 使用的 metrics 格式是 TFM (TeX Font Metrics)。

TrueType 的文件后缀是 .ttf，OpenType 的是 .ttf 和 .otf。METAFONT 虽然用矢量图形来定义字形，实际输出的却是一种点阵格式：PK (packed raster)。

上述字体按技术的先进性，从高到低的排序为：OpenType、TrueType、Type 1、Type 3、PK，我们应优先选用 OpenType 和 TrueType。

### 8.2.3 合纵连横

Adobe 收取的 Type 1 专利许可费一度十分昂贵，穷人们只好用免费的 Type 3。为了打破这种垄断，Apple 开发了 TrueType。1991 年 TrueType 发布之后，Adobe 随即公开了 Type 1 的规范，Type 1 字体从贵族堕落在平民，因而流行开来。

1980 年代中后期，Adobe 的大部分盈利来自于 PS 解释器的许可费。面对这种垄断局面，微软和 Apple 联合了起来。微软把买来的 PS 解释器 TrueImage 授权给 Apple，Apple 则把 TrueType 授权给微软。

微软得陇望蜀，又企图获得 AAT 的许可证，未遂。为了打破 Apple 的垄断，微软联合 Adobe 在 1996 年发布了 OpenType。Adobe 在 2002 年末将其字体库全面转向 OpenType。

上面这几出精彩好戏充分展示了商场上的勾心斗角、尔虞我诈，没有永恒的伙伴，只有永恒的利益。但它同时也告诉我们，市场竞争中受益的还是广大的消费者。

## 8.3 字体应用

PS 支持 Type 1 和 Type 3，而 PDF 除了这两种还支持 TrueType 和 OpenType。latex、DVI 浏览器、各种 driver 分别采用不同的字体技术。

### 8.3.1 DVI

latex 编译 L<sup>A</sup>T<sub>E</sub>X 源文件生成 DVI 时只需要 .tfm 文件，因为 DVI 并不包含字形信息，而只包含对字体的引用。DVI 浏览器显示 DVI 时一般使用 PK，它在系统中查找相应的 .pk 文件，若找不到就调用 METAFONT 在后台自动生成。

### 8.3.2 dvips

缺省情况下，dvips 也会查找 .pk，或调用 METAFONT 自动生成；然后把 PK 转换成包含点阵字体的 Type 3，它的参数 -D 可以用来控制该点阵字体的分辨率。用 ps2pdf 处理含 Type 3 的 PS 时，输出的自然是含 Type 3 的 PDF。

GSview 在低分辨率下可以很好地渲染 Type 3，Adobe Reader 或 Acrobat 却不能，因为它们使用的 Adobe Type Manager 不支持包含完整 PS 的 Type 3。含 Type 3 的 PDF 看起来会有些模糊，所以应尽量避免使用。

dvips 的另一个参数 -Ppdf 把 Type 1 嵌入生成的 PS，这样再 ps2pdf 就能生成含 Type 1 的 PDF。

dvips 不支持真正的 (native) TrueType，用户只能把 TrueType 先转成 PK 或 Type 1，这样绕了个弯效果总会打些折扣。

dvips 的字体详细使用方法可查阅其手册<sup>[1]</sup>第 6 章，此处不赘述。

### 8.3.3 dvipdfm(x)

dvipdfm 支持 PK 和 Type 1, 它可以用一个 `tlfonts.map` 文件建立 PK 文件和 Type 1 文件之间的映射, 这样生成的 PDF 用的就是 Type 1。dvipdfm 也不支持真正的 TrueType。

dvipdfmx 通过正确的设置可以使用真正的 TrueType, 它对中日韩等东亚文字的支持也较好, 所以它对我们来说是 Driver 的首选。

## 8.4 TrueType 字体安装配置

CJK 自带的 UTF-8 编码字体 gbsn 和 gkai 只包含 GB2312 字符集, 而 CTeX 只提供 GBK 编码字体, 因此中文用户通常需要自己安装配置 UTF-8 编码的 TrueType 字体。

在使用 TrueType 之前, 用户通常需要作以下准备工作:

1. 用转换程序 `ttf2tfm` 生成 TFM。
2. 配置字体定义文件 `.fd`。
3. 配置 `ttf2pk`, 因为 DVI 浏览器和 `dvips` 都会自动调用 `ttf2pk` 来生成 PK。
4. 配置 `dvipdfmx`。

### 8.4.1 目录和文件

通常每个发行包都会参照 TDS 建立自己的目录系统, 把各种文件发在固定的位置。比如 MiKTeX 顶层目录如下, 在本节后面的示例中我们将使用这些目录的缩写。

```
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
```

目录多了有个缺点，文件不知道放在哪里好。MiKTeX 中有的配置文件居然在四个目录下各有一份，实在是令人发指。幸好我们可以用下面的命令检查配置文件的具体名字和路径。

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

### 8.4.2 ttf2tfm

比如我们想把 `SimSun18030.ttc`（18030 字符集的新宋体）转换为 UTF8 编码的字体文件，我们需要执行以下步骤。

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

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

### 8.4.3 字体定义文件

字体定义文件将字体引用名和实际的字体文件联系起来，比如我们在 CJK 环境中引用 `usong` 时，系统将会找到并使用 `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`，在其它的发行包中可能是 `ttf2pk.cfg`。

`ttf2pk.ini` 中有一个 `.map` 文件列表，后者定义了 TrueType 应该按编码转为 PK 等信息。

比如下面这个文件列表会让 `ttf2pk` 读取 `foo.map` 和 `bar.map`。

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

如果系统找不到 `ttf2pk.ini`，它会缺省使用 `ttfonts.map`。

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

#### 8.4.5 配置 dvipdfmx

配置 `dvipdfmx` 是为了让 PDF 正确地嵌入 TrueType，否则生成的文件中的内容不能复制、粘贴。

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

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

## 参考文献

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

看什么看，没见过空白页？

# 跋

首先向一路披荆斩棘看到这里的读者表示祝贺，至少在精神上你已经成为一名合格的 L<sup>A</sup>T<sub>E</sub>Xer。从此你生是 L<sup>A</sup>T<sub>E</sub>X 的人，死是 L<sup>A</sup>T<sub>E</sub>X 的鬼。Once Black, never back。没有坚持到这里的同学自然已经重新投向“邪恶”的 MS Word，毕竟那里点个按钮就可以插入图形，点个下拉框就可以选择字体。

当然 L<sup>A</sup>T<sub>E</sub>Xer 也有简单的出路，就是只使用缺省设置，尽量少用插图；不必理会点阵、矢量，也不必理会 Type 1、Type 3、TrueType、OpenType。因为内容高于形式，你把文章的版面、字体搞得再漂亮，它也不会因此成为《红楼梦》；而《红楼梦》即使是手抄本，也依然是不朽的名著。

包老师曾经以为 L<sup>A</sup>T<sub>E</sub>X 和 Word 的关系就好像是《笑傲江湖》中华山的气宗和剑宗，头十年剑宗进步快，中间十年打个平手，再往后气宗就遥遥领先。至于令狐冲的无招胜有招，风清扬的神龙见首不见尾又是另一重境界，普通人恐怕只能望其颈背。

费尽九牛二虎之力熬到本文杀青的时候，才发现从前的想法很傻很天真。让我们挥一挥衣袖，不带走一片云，卧薪尝胆忍辱负重，耐心等待 X<sub>E</sub>L<sup>A</sup>T<sub>E</sub>X 和 Lua<sub>T</sub><sub>E</sub>X。