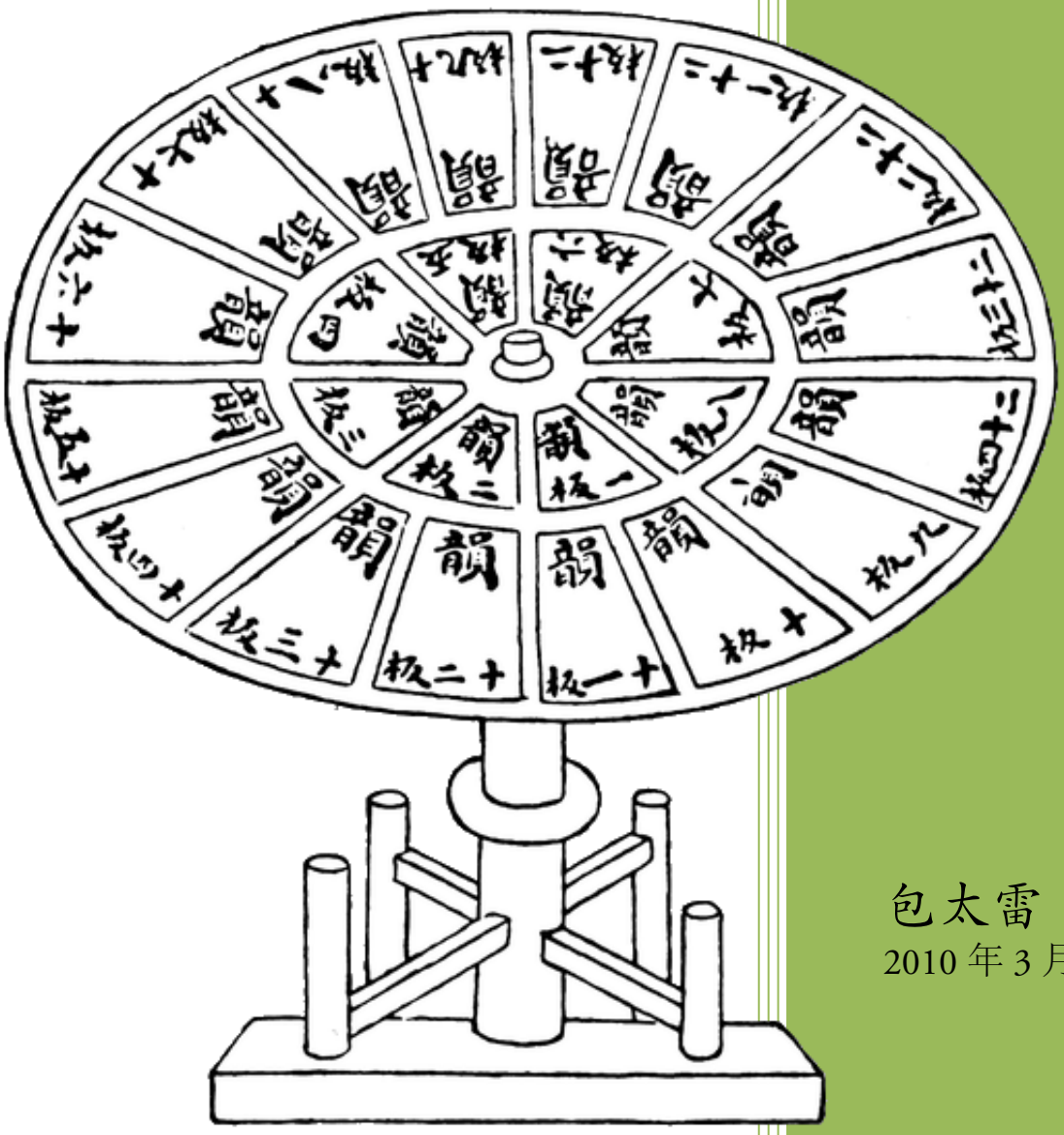
雷坛巨擘
扛鼎力作

LaTeX Notes

雷太赫排版系统简介

第二版 v2.0



包太雷

2010 年 3 月

LaTeX Notes

雷太赫排版系统简介

第二版 v2.0

包太雷

2010 年 3 月

Alpha Culture & Education

一品文化教育

Printed in the United States of America

To: Anna and Bella

广告位招租

目录

评论 xv

再版序 xvi

原版序 xviii

致谢 xx

1 简介 1
- 1.1 历史回顾 1
  - 1.1.1 页面描述语言 2
  - 1.1.2 标记语言 . . . 4
  - 1.1.3 TEX 家族 . . . 7
  - 1.1.4 小结 11
- 1.2 优点缺点 11
- 1.3 软件准备 12
- 1.4 学习方法 13

2 入门 15
- 2.1 Hello, World! 15
- 2.2 语法和结构 16
  - 2.2.1 语法 16
  - 2.2.2 物理结构 . . . 16
  - 2.2.3 逻辑结构 . . . 17
- 2.3 文字 19
  - 2.3.1 字符输入 . . . 19
  - 2.3.2 字体样式和大小 20
  - 2.3.3 换行、换页和断字 21
- 2.4 长度 22
- 2.5 对齐和间距 22
  - 2.5.1 段落对齐 . . . 22
  - 2.5.2 缩进和段间距 23
  - 2.5.3 行间距 23
- 2.6 特殊段落 24
  - 2.6.1 摘录 24
  - 2.6.2 原文打印 . . . 25
  - 2.6.3 脚注 25
  - 2.6.4 边注 26
  - 2.6.5 注释 26
- 2.7 列表 27
  - 2.7.1 基本列表 . . . 27
  - 2.7.2 其他列表 . . . 27
  - 2.7.3 定制列表 . . . 28
- 2.8 盒子 29
  - 2.8.1 初级盒子 . . . 29
  - 2.8.2 中级盒子 . . . 29
  - 2.8.3 高级盒子 . . . 30
- 2.9 交叉引用 30

3 字体 33
- 3.1 字符集和编码 33
- 3.2 字体格式 35
  - 3.2.1 点阵和矢量字体 35
  - 3.2.2 常见字体格式 35
  - 3.2.3 合纵连横 . . . 36
- 3.3 常见字体 37
- 3.4 字体的应用 37
  - 3.4.1 早期技术 . . . 37
  - 3.4.2 XeTeX 38
- 3.5 中文解决方案 39

4 数学 43
4.1 数学模式 43
4.2 基本元素 44
4.2.1 希腊字母 . . . 44
4.2.2 上下标和根号 45
4.2.3 分数 45
4.2.4 运算符 45
4.2.5 箭头 47
4.2.6 注音和标注 . 47
4.2.7 分隔符 48
4.2.8 省略号 49
4.2.9 空白间距 . . . 49
4.3 矩阵 49
4.4 多行公式 50
4.4.1 长公式 50
4.4.2 公式组 51
4.4.3 分支公式 . . . 51
4.5 定理和证明 51
4.6 数学字体 53

5 插图 55
5.1 图形概览 55
5.1.1 图形格式 . . . 55
5.1.2 Driver 的口味 56
5.1.3 图形优化 . . . 57
5.1.4 图形转换和处理 61
5.2 插入图形 63
5.2.1 范围框 63
5.2.2 基本命令 . . . 64
5.2.3 图形操作 . . . 64
5.2.4 文件名和路径 66
5.2.5 figure 环境 . . 67
5.2.6 插入多幅图形 67
5.3 矢量绘图 70
5.3.1 色彩模型 . . . 70
5.3.2 绘图工具概览 73

6 Metapost 75
6.1 准备工作 75
6.2 基本图形对象 76
6.2.1 点和直线 . . . 76
6.2.2 预定义形状 . 77
6.2.3 曲线 77
6.3 图形控制 78
6.3.1 线型 78
6.3.2 箭头 78
6.3.3 彩色和填充 . 78
6.4 图形变换 79
6.5 标注 80
6.6 编程 82
6.6.1 数据类型和变量 82
6.6.2 数学运算 . . . 83
6.6.3 循环 83

7 PSTricks 85
7.1 准备工作 85
7.2 基本图形对象 87
7.2.1 点和直线 . . . 87
7.2.2 矩形和多边形 87
7.2.3 圆、椭圆、圆弧、扇形 . 88
7.2.4 曲线 88
7.2.5 网格和坐标轴 89
7.3 图形控制 90
7.3.1 线宽和线型 . 90
7.3.2 箭头 90
7.3.3 颜色和填充 . 90
7.3.4 全局设置 . . . 91
7.4 图形变换 91
7.5 标注 92

8 PGF 95
8.1 准备工作 95
8.2 基本图形对象 96
8.2.1 直线和矩形 . 96
8.2.2 圆、椭圆、弧 97
8.2.3 曲线 97
8.2.4 网格 98
8.3 图形控制 98
8.3.1 箭头 98
8.3.2 线宽和线型 . 98
8.3.3 颜色和填充 . 99
8.3.4 渐变和阴影 . 99
8.3.5 样式 99
8.4 图形变换 100
8.5 示意图 101

8.5.1 节点 101
8.5.2 流程图 101
8.5.3 树 102
8.5.4 预定义节点形状 102
8.6 编程 104
8.6.1 循环语句 . . . 104
8.6.2 数据图 104

9 表格 107
9.1 简单表格 107
9.2 宽度控制 108
9.3 跨行跨列 110
9.4 数字表格 111
9.5 长表格 112
9.6 宽表格 114
9.7 彩色表格 117

10 结构 121
10.1 长文档 121
10.2 标题 122
10.3 目录 122
10.4 参考文献 124
10.4.1 `thebibliography` 124
10.4.2 BibTeX 125
10.4.3 Natbib 128
10.5 索引 129
10.6 超链接 130
10.7 结构名 131

11 布局 133
11.1 页面尺寸 133
11.1.1 普通青年 . . . 133
11.1.2 文二青年 . . . 134
11.1.3 尺寸详解 . . . 135
11.2 页面样式 137
11.3 分栏 141
11.4 分页 143

12 应用 145
12.1 幻灯 145
12.2 海报 145
12.3 书信 145
12.4 简历 145
12.5 棋谱 145

A 软件安装 147

B 印刷简史 149
B.1 传统印刷 149
B.1.1 凸版印刷 . . . 150
B.1.2 凹版印刷 . . . 152
B.1.3 平版印刷 . . . 153
B.1.4 孔版印刷 . . . 154
B.2 传统排版 155
B.2.1 手工排版 . . . 155
B.2.2 机械排版 . . . 155
B.2.3 照相排版 . . . 156
B.3 数字印刷 157
B.3.1 光电打印机 . 158
B.3.2 喷墨打印机 . 158
B.3.3 撞击式打印机 159
B.3.4 热学打印机 . 160
B.3.5 绘图仪 160
B.4 数字排版 160

再版跋 162

原版跋 163

索引 164
人物索引 165
组织机构索引 168

图目录

2.1 编译和格式转换 16

5.1 缺省输出尺寸 57
5.2 强制放大输出尺寸 57
5.3 原始图形 59
5.4 改尺寸 59
5.5 改分辨率 59
5.6 改尺寸和分辨率 59
5.7 色彩深度 60
5.8 反清复明 68
5.9 清明 68
5.10 反复 68
5.11 反清复明 69
5.12 反清复明 70
5.13 RGB 模型 71
5.14 CMYK 模型 71
5.15 选色 71

10.1 BibTeX 的编译 127
10.2 索引的编译 130

11.1 页面尺寸 135

B.1 《金刚经》，雕版，868 151
B.2 Gutenberg Bible, letterpress, 1455 151
B.3 St. Christopher, engraving by Dürer, 1521 152
B.4 Woman in Cafe, drypoint by Lesser Ury 152
B.5 The Soldier and his Wife, etching by Hopfer, 1500 153
B.6 Georgiana Cavendish, Duchess of Devonshire, mezzotint 153
B.7 Sleep of Reason Produces Monsters, etching, aquatint, and drypoint by Goya, 1799 154
B.8 The Bath, drypoint and aquatint by Cassatt, 1890 154

表目录

1.1 数字排版工作流程及主要工具 2
1.2 TEX 发行版与编辑器 13

2.1 文档类常用选项 17
2.2 特殊符号和预定义字符串 20
2.3 注音符号 20
2.4 字体样式 21
2.5 字体相对尺寸 21
2.6 常用长度单位 22
2.7 计数器显示格式 26

3.1 常见字体 37

4.1 简单数学公式的输入 44
4.2 希腊字母 44
4.3 箭头 47
4.4 数学注音符号 47
4.5 长标注符号 48
4.6 空白间距 49
4.7 数学字体 53

5.1 图形操作选项 65

6.1 label 命令的方位 80
6.2 METAPOST数学函数 84

7.1 PSTricks 辅助宏包 85
7.2 rput 命令的参考点 92
7.3 uput 命令的角度参数 93

9.1 长表格 113
9.2 主流英语词典 116

10.1 参考文献引用样式选项 128

11.1 LaTeX 页面样式 137
11.2 页眉和页脚常用宏变量 138

A.1 软件链接 147

例目录

2.1 Hello, Wolrd! 15
2.2 层次结构 18
2.3 划线和减号 21
2.4 字体强调和下划线 22
2.5 段落对齐方式 23
2.6 行距命令 24
2.7 行距环境 24
2.8 摘录环境 24
2.9 原文打印 25
2.10 脚注 25
2.11 边注 26
2.12 基本列表 27
2.13 压缩列表 28
2.14 行间列表 28
2.15 定制列表 29
2.16 `mbox`和 `fbox` 29
2.17 `makebox`和 `framebox` 30
2.18 `parbox` 和 `minipage` 30
2.19 交叉引用 31
3.1 XeLaTeX 字体设置 39
3.2 xeCJK 40
4.1 数学模式 44
4.2 上下标和根号 45
4.3 分数 45
4.4 小运算符 45
4.5 大运算符 46
4.6 积分变量 46
4.7 多重积分 47
4.8 可扩展箭头 47
4.9 分隔符 48
4.10 省略号 49
4.11 矩阵 50

4.12 更多矩阵 . . . 50
4.13 行间矩阵 . . . 50
4.14 无对齐长公式 . . . 51
4.15 对齐长公式 . . . 51
4.16 无对齐公式组 . . . 51
4.17 对齐公式组 . . . 51
4.18 分支公式 . . . 52
4.19 定制定理类环境 . . . 52
4.20 使用定理类环境 . . . 52
4.21 证明 . . . 53
5.1 ImageMagick 尺寸操作 . . . 61
5.2 ImageMagick 色深操作 . . . 62
5.3 图形缩放 . . . 65
5.4 图形旋转 . . . 66
5.5 插图文件名和路径 . . . 67
5.6 figure 环境 . . . 67
5.7 并排摆放，共享标题 . . . 68
5.8 并排摆放，各有标题 . . . 68
5.9 并排摆放，共享标题，各有子标题 . . . 69
5.10 改进的子图方法 . . . 70
5.11 自定义颜色 . . . 72
5.12 彩色文字 . . . 73
5.13 彩色盒子 . . . 73
6.1 METAPOST源文件 . . . 76
6.2 METAPOST点和直线 . . . 77
6.3 METAPOST预定义形状 . . . 78
6.4 METAPOST曲线 . . . 79
6.5 METAPOST线型 . . . 80
6.6 METAPOST箭头 . . . 80
6.7 METAPOST彩色 . . . 81
6.8 METAPOST填充 . . . 81
6.9 METAPOST图形变换 . . . 82
6.10 METAPOST标注 . . . 83
6.11 METAPOST循环 . . . 84
7.1 pst-pdf 宏包 . . . 86
7.2 preview 宏包 . . . 86
7.3 PStricks 点和直线 . . . 87
7.4 PStricks 矩形和多边形 . . . 87
7.5 PStricks 圆、椭圆、圆弧、扇形 . . . 88
7.6 PStricks 曲线 . . . 88
7.7 PStricks 网格 . . . 89
7.8 PStricks 坐标轴 . . . 89
7.9 PStricks 线宽和线型 . . . 90
7.10 PStricks 箭头 . . . 90

7.11 PStricks 彩色 . . . 91
7.12 PStricks 填充 . . . 91
7.13 PStricks 平移和旋转 . . . 92
7.14 PStricks 标注 . . . 93
8.1 制作独立的 PGF 图形文件 . . . 96
8.2 PGF 直线和矩形 . . . 96
8.3 PGF 圆、椭圆、弧 . . . 97
8.4 PGF 曲线 . . . 97
8.5 PGF 网格 . . . 98
8.6 PGF 箭头 . . . 98
8.7 PGF 线宽和线型 . . . 99
8.8 PGF 颜色和填充 . . . 99
8.9 PGF 阴影 . . . 100
8.10 PGF 全局样式 . . . 100
8.11 PGF 局部样式 . . . 100
8.12 PGF 图形变换 . . . 101
8.13 PGF box 样式 . . . 101
8.14 PGF 流程图 . . . 102
8.15 PGF 又一个流程图 . . . 102
8.16 PGF 好大一棵树 . . . 103
8.17 PGF 预定义节点形状 . . . 103
8.18 PGF 预定义正多边形节点 . . . 104
8.19 PGF 循环语句 . . . 104
8.20 PGF 函数图 . . . 105
9.1 简单表格 . . . 107
9.2 浮动三线表 . . . 108
9.3 控制列宽 . . . 109
9.4 控制列宽和横向对齐 . . . 110
9.5 控制表格宽度 . . . 110
9.6 跨列表格 . . . 111
9.7 数字表格 . . . 112
9.8 彩色表格 . . . 117
9.9 彩色表格 . . . 118
10.1 拆分长文档 . . . 122
10.2 标题 . . . 123
10.3 目录 . . . 123
10.4 自定义浮动环境 . . . 124
10.5 thebibliography 环境 . . . 124
10.6 BibTeX 数据 . . . 126
10.7 子文档参考文献的编译 . . . 128
10.8 各种引用模式下的引用命令效果 . . . 129
10.9 索引 . . . 129
10.10 hyperref 命令 . . . 130
10.11 url 和 href 命令 . . . 130

10.12标准文档类结构名重定义 131
10.13hyperref 宏包结构名重定义 132
11.1 自定义页面样式 138
11.2 headings 样式 139
11.3 myheadings 样式 139
11.4 定制左右标记 140
11.5 fancyhdr 宏包 140
11.6 定制章节标记 141
11.7 multicol 宏包 142
11.8 特殊浮动体 142

评论

包太雷同志是一个高尚的人，一个纯粹的人，一个有道德的人，一个脱离了低级趣味的人，一个有益于人民的人。

— 白求恩

一个人偶尔雷人容易，难的是一辈子雷人。

— 雷锋

书中自有颜如玉。

— 如花

包老师帅得惊动了党中央。

— 犀利

LXJX。

— 蓝翔技校

再版序

转眼间 lnotes 已经两岁了。在这两年间，它在雷友们的帮助下改正了许多缺点并茁壮地成长着。随着 LaTeX 技术的进步，包老师感觉到零敲碎打缝缝补补已经不适应新时期的发展；lnotes 需要洗心革面重新做人，以求得人民群众的谅解。因为太懒，这个动手改版的日期被一再推迟。然而亡羊补牢，亦未为迟。

新版改进

新版本力求实现以下几个目标：

1. 系统性；结构完整，脉络清晰。
2. 层次性；详略得当，重点突出。
3. 进步性；技术先进，内容创新。
4. 一致性；前后呼应，风格统一。

古人云：知易行难。古人又云：法乎其上，得乎其中。为实现这些目标，本文作出以下具体调整：

1. 全文按出版业惯例分为三大部分：前置部分，包括封面、标题、版权、献辞、目录、序、致谢等；主体部分包括九章；后置部分包括跋、附录、索引等。
2. 第一章简介，历史回顾部分有较大扩充。
3. 第二章入门，结构有所调整，重组了对齐和间距、特殊段落、列表等节，内容也有增强。
4. 第三章字体，介绍电脑字体相关概念，字符集、编码、字体格式等，以及字体在 LaTeX 中的应用。由于 XeTeX 日趋成熟，新字体技术方案得到广泛应用；从前的一些中文解决方案和字体安装配置方法已经过时。本章由原第七章中文和第八章字体压缩合并而来，增加了 XeTeX 相关内容。

5. 第四章数学，结构基本不变，补充了部分示例的代码。
6. 第五章插图，介绍图形格式及其转换，怎样插入图形，矢量绘图。增加了色彩模型介绍；若干小节标题略作调整，更加一致。
7. METAPOST, PSTricks, PGF 等绘图工具的介绍因篇幅较长，拆分为三个独立章节，内容亦有所增强和扩充。
8. 第九章表格，增加了数字表格和宽表格两节，宽度控制和彩色表格有所增加和扩充。
9. 第十章结构，介绍文档结构，标题、目录、长文档、参考文献、索引、超链接等。
10. 第十一章布局，介绍页面尺寸，页面样式以及左右标记、章节标记等的定制，多栏、分页等。
11. 第十二章应用，包括幻灯、海报、书信、简历、棋谱等内容。
12. 附录 A 软件安装，未完成。
13. 附录 B 印刷简史，有图有真相。
14. 索引，包括人物、组织机构索引。

体例

还没总结好。

1. 人名，西方人名正文中一般用原文，索引中加中文翻译。中国人和日本人正文中用中文，索引中加英文翻译。正文中人名第一次出现时附生卒年份。
2. 组织、机构、公司、学校等，著名的正文中用中文，比如惠普、微软，索引中加英文。缩写，著名的正文中直接用，比如 MIT, IBM, ISO，索引中有全名；一般的先写中文名，括号内提供英文名和缩写。
3. 英文大小写，组织、机构、公司、学校等，一般首字母大写；其他词汇全部小写。
4. 标点符号，中文间用全角，西文间用半角。
5. 命令行程序、宏包、LaTeX 命令和环境等用等宽字体。

原版序

满纸荒唐言，一把辛酸泪！都云作者痴，谁解其中味？[1]

— 曹雪芹

最早听说 LaTeX 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[2]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 LaTeX 而走火入魔。

大约是 2005 年底，看了一下 lshort，用 LaTeX 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 LaTeX 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 LaTeX 的威力。

就这样接触了两三年，总算略窥门径，感觉 LaTeX 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。

日前老妻吵着要学 LaTeX，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。

[1] 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。

[2] 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。

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 LaTeX 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[3]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[4] 。

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和整理这份笔记。

[3]此处用法循阿扁古例。

[4]huang.xingang@gmail.com

致谢

在本文的写作过程中，我得到了众多网友的帮助和指点，各位反动学术权威的关心和鼓励。没有你们的帮助，包老师形只影单单枪匹马马不停蹄也难以完成这件超出本人能力的事情。

在此包老师依据我国法律[5]，首先郑重感谢党和政府的栽培，国家和人民的养育，以及有关部门的领导。感谢铁岭 TV，辽宁 TV，将来还有可能感谢 CCTV。

其次将网友们的名单公诸于众，以彰显社会良知、公民勇气。以下排名不分先后，其中多数网友来自水木清华 BBS TEX 版和 CTEX 论坛。

careworn@smth.org
Dieken@smth.org
donated@smth.org
hkkhhk
Hongdong Ji
IMB@smth.org
jjgod@smth.org
Kov Chai
Langpku@smth.org
LittleLeo@smth.org
meteorrain@smth.org
milksea@smth.org
PaladinHL@smth.org
PiscesGold@smth.org
primenumber@smth.org
snoopyzhao@smth.org
tex@smth.org
Xiao Zigang
Xubuntu@smth.org
yakun@smth.org
yli@smth.org
yyzz11@smth.org
张晓南
贾朋

也借机感谢一下 4.80[6]前站长 Jonny 和水木清华 BBS 前站长 Leeward。十余年前，在我开始浸淫于电脑网络时，这两位高手对我有很大的启发和影

[5] 《中华人民共和国感谢法》，2010 年 3 月 12 日。

[6] 166.111.4.80 是清华大学校园网早期一个重要的 FTP 站点。

响。虽然两位高人淡出公众视野已久，但是他们为人民服务的精神却依然值得我们缅怀与尊敬。

最后还要感谢家人的理解和支持。老妻把她的博士论文给我当作学习LaTeX的试验品；大女儿把她的玉照给我当作插图样板；小女儿把她的名字给我放在献辞页。

广告位招租

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》

排版是人类生活中一项很重要的工作，也是传统印刷和电脑出版的核心活动。在一定的版面内摆放不同型态的对象（如数字、文字、表格和图形等），以合适的方法表现渲染它们，这个过程就是排版。

排版的版面可以是有固定尺寸的印刷品，也可以是较为灵活的电脑软件、电子文档，还可以是狂野奔放的网页。

1.1　历史回顾

排版按照历史时期和技术方法可以划分为传统排版和数字排版两大类。数字排版的一个重要概念是光栅图像处理器 (raster image processor, RIP)。RIP 出现之前的印刷排版历史可参阅 附录 B。

RIP 生成的点阵图像可以输出到激光照排机或直接制版机等制版设备，也可以输出到打印机；它的输入可以是页面描述语言 (page description language, PDL)，也可以是和输出设备分辨率不同的点阵图像。

RIP 可以是硬件、固件或软件。硬件 RIP 用于高档排版设备，1976 年莫诺公司 (Monotype Corp.) 的激光照排机 Lasercomp 就配有硬件 RIP。固件 RIP 在打印机内置微处理器上运行，每一台 PostScript (PS) 打印机都配有固件 RIP，比如 1985 年苹果公司的 LaserWriter。最早的软件 RIP 是 1986 年的 Ghostscript。

顾名思义，页面描述语言是用来描述待输出页面的，它比二进制的图像数据高级一点，但是正常人类看起来还是会很费劲。所以人们要在它前面再加一层标记语言 (markup language) 。图像数据、页面描述语言、标记语言的关系大致可以比喻为机器码、汇编语言、高级语言。而在排版领域，TEX 是最精确最强大的标记语言，可谓公鸡中的战斗机。

数字排版的工作流程和主要工具见 表 1.1。本节以下部分将分别回顾常见的页面描述语言、标记语言和 TEX。

表 1.1: 数字排版工作流程及主要工具

标记语言	→	PDL	→	RIP	→	输出设备
troff 系列		PS		硬件 RIP		激光照排机
SGML 系列		PDF		固件 RIP		直接制版机
TEX 系列		DVI		软件 RIP		打印机

1.1.1 页面描述语言

PostScript

最早的行式打印机只能打印字符，后来的针式打印机可以用点阵的方式画出字符，也可以画出粗糙的图形。当时矢量图只能用绘图仪来打印。

1969 年施乐推出首台激光打印机之后，就一直在想办法精确描述页面图像，从而结合点阵打印机和绘图仪的优点，同时打印高质量的图形和文字。1975 年 Robert F. Sproull [1] 主持开发了一种格式 Press，后来用于施乐的 Xerox Star 电脑 (一种个人电脑的雏形) 。但是 Press 只是一种格式而不是语言，所以施乐启动了 InterPress 研究计划。

1976 年，埃文斯 · 苏泽兰公司 (Evans & Sutherland) 的 John E. Warnock (1940–) [2] 正在酝酿一种图形设计语言，也就是后来的 PostScript。1978 公司想让他从旧金山分部搬到犹他州总部，他不想搬家就跳槽到了施乐。

埃文斯 · 苏泽兰的两位创始人 David C. Evans (1924–1998) [3] 和 Ivan E. Sutherland (1938–) [4] 都是犹他大学的教授，也是 Warnock 的博士导师。他们

[1] 1967 年哈佛大学物理学士，斯坦福大学计算机系 1970 年硕士，1977 年博士。先后供职于施乐和 Sun，中间还当过一阵卡耐基梅隆大学计算机系副教授。

[2] 犹他大学数学系 1961 年学士，1964 年硕士，1969 年电脑博士。

[3] 犹他物理系 1949 年学士，1953 年博士。毕业后加入本迪克斯公司 (Bendix Corp.) ，1962 年跳槽到犹他电子系，1965 年创立犹他计算机系。

[4] 1959 年卡耐基梅隆电子学士，1960 年加州理工学院电子硕士，1963 年 MIT 电脑博士。1963 年国家安全局 (National Security Agency, NSA) 中尉，1964 年国防部高等研究计划局

对施乐挖走得意门生耿耿于怀。1980 年 Sutherland 从施乐挖走另一位弟子 Sproull，成立了一家咨询公司苏泽兰 · 斯普罗及其同伙 (Sutherland, Sproull and Associates) 。1990 年这家公司被 Sun 收购，师徒两位都接着给 Sun 工作。

在施乐这边，Warnock 和 Martin Newell开发了新的图形系统 JaM (John and Martin)，它后来被合并到 InterPress 中去。这两位还开发过另一个系统 MaJ。

1982 年，Warnock 和同事 Charles M. Geschke (1939–) [5] 一起离开施乐，创立了 Adobe。Newell 后来也加入了 Adobe，于是施乐的 InterPress 胎死腹中。

1984 年 Adobe 发布 PostScript 后不久，乔帮主 (Steve Jobs, 1955–) 跑来参观，并建议用它来驱动激光打印机。次年，苹果推出了配备 PS 解释器的 LaserWriter。之后 Adobe 分别于 1991 和 1997 年推出了 PostScript 2 和 PostScript 3，这三代 PostScript 都是当时的流行标准。据说 1980 年代 Adobe 的收入多数来自 PS 解释器的许可费。

1990 年代后期，廉价喷墨打印机的出现使得 PostScript 逐渐式微，因为 PS 解释器对它们毕竟是一个成本负担，另外 PostScript 过于复杂，对微处理器和内存要求都很高。

PDF

1993 年，Adobe 推出了另一种格式 portable document format (PDF) ，它在 2008 年成为开放标准 ISO 32000。除了开放，PDF 比起 PostScript 还有其他一些优势：

- PDF 基本上是 PostScript 的一个子集，因此更轻便。
- PDF 支持更先进的字体，具体见第三章。
- PDF 支持透明图形，还支持动画。
- PDF 支持加密等安全特性。

虽然拥有上述优势，PDF 最初的推广却并不顺利，因为其读写工具 Acrobat 太贵。后来 Adobe 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 PostScript，成为网络时代电子文档的新标准。

(Defense Advanced Research Projects Agency，DARPA) 信息处理主管。1966 年哈佛电子系副教授，1968 年与其学生 Sproull 发明了虚拟现实技术。1968 年犹他计算机系教授，并与同事 Evans 创立埃文斯 · 苏泽兰。1974 年创立加州理工计算机系。1988 年获图灵奖 (Turing Award) ，1998 年获冯 · 诺依曼奖 (IEEE John von Neumann Medal) 。

[5] 泽维尔大学 (Xavier University) 1962 年西方古典学士，1963 年数学硕士，卡耐基梅隆 1972 年电脑博士。

其他页面描述语言

其他页面描述语言还有:

- 爱普生打印机标准码 (Epson standard code for printers, ESC/P) ，主要用于针式打印机。
- 惠普的打印机命令语言 (printer command language, PCL) ，主要用于喷墨打印机。
- 惠普图形语言 (HP graphics language, HPGL) ，主要用于绘图仪。
- TEX 家族的设备独立文件格式 (device independent file format, DVI) ，详见 1.1.3 节。
- 微软的 XML 纸张规范 (XML paper specification, XPS) 。2009 年，基于 XPS 的 Open XPS 被欧洲计算机制造商协会 (European Computer Manufacturers Association) 批准为 ECMA 标准。

1.1.2 标记语言

troff 系列

1964 年，MIT 的 Jerome H. Saltzer (1939–) [6] 在参与开发第一个分时操作系统 compatible time-sharing system (CTSS) 时，写了一个文本排版程序 RUNOFF。

后来贝尔实验室的 Robert H. Morris [7] 把 RUNOFF 移植到 GE 635 上，改名为 roff。1969 年，Malcolm D. McIlroy (1932–) [8] 把 roff 用 BCPL 语言重写，移植到 DEC PDP-7。

1971 年，一小撮 Unix 开发人员想把他们的电脑升级到 PDP-11，谎称升级的原因是要给贝尔实验室的东家 AT&T的专利部门做一个文件排版系统。技术员 Joseph F. Ossanna (1928–1977) [9] 就把 roff 改写为 nroff。后来他们弄到一台王安公司 (Wang Laboratories) 的排版机 WANG Graphic Systems CAT，Ossanna 又把 nroff 用 PDP-11 的汇编语言改写为 troff。

[6] MIT 电子系 1961 年学士，1963 年硕士，1966 年博士，同年留校任教，1995 年退休。他是一位从一而终的典范。

[7] 哈佛数学系 1957 年学士，1958 年硕士。1960 年加入贝尔实验室，1986 年跳槽到 NSA，曾任 NSA 电脑安全中心首席科学家，1995 年退休。他为 Unix 写了数学库函数。1988 年他儿子 Robert T. Morris (1965–) 在康奈尔大学念研究生时，联网到 MIT 释放了第一个电脑蠕虫；被判三年监禁，400 小时社区劳动，罚款一万。

[8] 1954 年康奈尔工程物理学士，1959 年 MIT 应用数学博士。1958 年加入贝尔实验室，1997 年退休。曾任图灵奖委员会主席。

[9] 1952 年毕业于韦恩州立大学 (Wayne State Univesrity) 。

领导说，小同志啊，现在 C 语言是三个代表的。Ossanna 没日没夜地把它用 C 又写了一遍，7000 行不带注释的程序。领导又说，小同志啊，CAT 已经不时髦了，我们要高瞻远瞩放眼 21 世纪。Ossanna 又去加班，累倒在工作岗位上，终年 49 岁。据说他在弥留之际的遗言是，“同志们，不要管我，抢救公社的……要紧！”包子曰：四流大学小本在大公司里真的很难混啊。

关键时刻，白求恩的老乡 Brian W. Kernighan (1942–) [10] 挺身而出，把 troff 改写成独立于排版机的 ditroff，还发表了一篇技术报告，*A Typesetter-independent TROFF*。包子曰：编程、灌水两手抓，两手都要硬。

troff 包含一系列命令，用来设置字体、间距、段落、旁注、脚注等，能够将字符在页面上任意定位，甚至重叠。当遇到复杂任务时，人们还可以利用 Unix 的管道功能，将 troff 和其他程序结合使用。

1990 年代贝尔实验室大分家，Unix 部分卖给了 Novell，其他部分大多重组进朗讯科技。troff 也就花落几家，重组后的贝尔实验室一份，朗讯软件一份，Sun 的创始人之一 Bill N. Joy (1954–) [11] 一份，GNU也搞了一个 groff。人心散了，队伍不好带了。

目前 troff 还是 Unix 手册的缺省文件格式，但是它在排版方面的领地已经基本被 LaTeX 和一些所见即所得软件瓜分。

SGML 系列

1969 年，IBM 的 Charles F. Goldfarb [12] 和同事 Edward Mosher, Raymond Lorie 发明了通用标记语言 (generalized markup language, GML)，GML 其实是他们三人姓氏的首字母。

GML 把结构、内容、格式分开，这样作者就可以专心写作。另外还可以为输出设备指定各自的特性文件 (profile)，从而实现设备独立。

1978 年，Goldfarb 等开始改进 GML。1986 年标准通用标记语言 (standard generalized markup language, SGML) 成为国际标准 ISO 8879。SGML 提出了条目、标签、元素、属性、实体等语法概念，还增加了有效性检验机制。

[10] 1964 年多伦多大学工程物理学士，1969 年普林斯顿大学电子博士。2000 年离开贝尔实验室，加入普林斯顿。1977 年与同事 Alfred V. Aho (1941) , Peter J. Weinberger一起发明了 AWK 语言。1978 年与 C 语言发明人 Dennis M. Ritchie (1941–) 合写了 *The C Programming Language*。1990 年与同事 Robert Fourer, David M. Gay发明了 AMPL 语言。

[11] 1975 年密歇根大学电子学士，1979 年伯克利大学电脑硕士。他开发了 vi 编辑器、C Shell (csh)，还参与了 BSD Unix、网络文件系统协议 (Network File System, NFS) 的开发。

[12] 1960 年哥伦比亚学院文学学士，1964 年哈佛法学博士。做了几年律师后于 1967 年加入 IBM，不知何年离开 IBM 又变回律师。

但是因为 SGML 过于复杂，一直没有流行起来，只有一些大公司或政府部门在使用。

1989 年，欧洲核子研究中心 (European Organization for Nuclear Research, CERN) 的 Tim J. Berners-Lee (1955–) [13] 设计了超文本标记语言 (HyperText Markup Language, HTML)，并写了浏览器和服务器软件。他认为 HTML 是一种 SGML 的应用。Berners-Lee 被称为万维网 (world wide web) 之父。1994 年，Berners-Lee 纠集欧盟委员会和 DARPA，在 MIT 创建了万维网联盟 (World Wide Web Consortium, W3C)。

1993 年，HTML 成为互联网工程任务组 (Internet Engineering Task Force, IETF) 的建议标准。1995 年 IETF 发布了 HTML 2.0。1997 年，HTML 3.2 成为 W3C 推荐标准，同年升级到 4.0，次年升级到 4.01。

HTML 擅长表现布局、外观，但是缺乏对内容数据的关怀，所以人们从 1990 年代中期就开始寻求面向数据交换的新格式。1998 年 W3C 发布了在 SGML 基础上简化来的扩展标记语言 (Extensible Markup Language, XML)，2004 年升级为 XML 1.1。

从那时起，基于 XML 的新名词儿如雨后春笋不断涌现。其中和排版关系比较大的是 1998 年 W3C 发布的数学标记语言 (mathematical markup language, MathML)，2003 年升级为 MathML 2.0。

1991 年豪电脑 (Hal Computer Systems) 和欧莱利出版公司 (O'Reilly Media) 推出了 DocBook，一种面向技术文档的标记语言。最初它算是 SGML 的应用，XML 出现后改换门庭，最新版本是 2009 年的 5.0。它采用了显示中性的方法，用它编写的的内容在出版时可以选择 HTML, PDF, CHM [14] 等多种格式，而用户无须修改源码。

Scribe

1980 年，卡耐基梅隆的 Brian K. Reid (1949–) [15] 提交了他的博士论文 *Scribe: A Document Specification Language and its Compiler*。1981 年，Reid 鼓动 Goldfarb 一起参加了一个会议，Goldfarb 发现 Scribe 和 SGML 在几处重要概念上是相似的，可谓英雄所见略同。

[13] 1976 年牛津大学物理学士。1980 年成为 CERN 合同工，1984 年转正。2004 年起任南安普敦大学电子系主任。

[14] Compiled HTML Help，微软的一种帮助文件格式，它对 HTML 进行了索引、压缩。

[15] 1970 年马里兰大学物理学士，1980 年卡耐基梅隆电脑博士。同年加入斯坦福，1987 年终身教职被拒，跳槽到 DEC。1999 年跳到贝尔实验室，2001 年跳到卡耐基梅隆。2002 年跳到 Google，2004 年 Google 首次公开募股 (IPO) 前被解雇，原因是他 too old。按 IPO 的价格，他损失了 1000 万美元左右，愤而起诉，官司好像还没打完。

Reid 在毕业时把 Scribe 卖给了一位教授 Michael I. Shamos (1947–) [16] 的统一逻辑公司 (Unilogic) 。Shamos 跟卡耐基梅隆为 Scribe 的知识产权打了几年官司，然后给它装上时间炸弹迫使用户交钱。Richard M. Stallman (1953–) [17] 谴责说这是对程序员精神的背叛，是对人类的犯罪。Scribe 后来无疾而终。

1.1.3 TEX 家族

包子曰：自施乐以降，豪杰并起，跨州连郡者不可胜数。初 SGML 名微而众寡，然遂能克 troff，以弱为强者，非惟天时，抑亦人谋也。今 SGML 已拥百万之众，挟互联网而令诸侯，此诚不可与争锋。土坯据有 PDL，已历三世，国险而民附，贤能为之用，此可以为援而不可图也。纳德将军既帝室之胄，信义著于四海，总揽英雄，思贤如渴，若跨有公式、算法，保其岩阻，西和 DocBook，南抚 Scribe，外结好土坯，内修政理；天下有变，则命一上将将公式之军以向 AMS、SIAM，将军身率算法之众出于 TUG，百姓孰敢不箪食壶浆，以迎将军者乎？诚如是，则霸业可成，雷太赫可兴矣。

引擎

谈到 TEX，人们首先会想起 Donald E. Knuth (1938–) [18]。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，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，这就是 *The Art of Computer Programming* [19]。

[16] 1968 年普林斯顿物理学士，1970 年瓦沙学院 (Vassar College) 物理硕士，1972 年美国大学 (American University) 技术管理硕士，耶鲁大学计算机系 1973, 74 年双硕士，1978 年博士。1981 年杜肯大学 (Duquesne University) 法学博士。有冇搞错，七个学位。1975 年加入卡耐基梅隆，历任数学、统计、电脑等系科助理教授、客座教授、图书馆馆长、研究所主任等职位。2001 年香港大学电子系访问教授。

[17] 1971 年在哈佛念大一时，高分通过了号称全美大学本科最难的数学课 Math 55。这门课的学生高考都是满分，最后只有四分之一能及格，通过者包括比尔 · 盖茨 (1955) 。1974 年获物理学士后，入 MIT 念博士，后为专心当黑客而退学。1970 年代开发了 Emacs, 1983 年创立 GNU，1985 年创立自由软件基金会 (Free Software Foundation, FSF)

[18] 1960 年凯斯工学院数学学士，因为长得帅同时获赠硕士，在校期间曾加入黑社会外围组织 ΘX。1963 年加州理工数学博士，同年留校任教。1968 年跳槽到斯坦福，1974 年获图灵奖，1992 年退休，1995 年获冯 · 诺依曼奖。

[19] 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。

1976 年，当 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；而当时数字排版刚刚兴起，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排版系统，这就是 TEX。

1978 年 TEX 第一版发布后好评如潮，Knuth 趁热打铁在 1982 年发布了第二版，1989 年发布的 TEX 3.0 将 7 位字符改为 8 位。之后 Knuth 宣布除了修正漏洞停止 TEX 的开发，因为它已经很稳定，而且他要集中精力完成那本巨著的后几卷。

从那时起，每发布一个修正版，版本号就增加一位小数，趋近于 π；当前版本是 2008 年的 3.1415926。他的另一个软件 METAFONT的版本号趋近于 e，目前是 2.718281。Knuth 希望在他离世时，TEX 和 METAFONT的版本号永远固定下来，从此人们不再改动他的代码。

Knuth 在软件工程方面也独树一帜。一般认为，编程语言大致可以划分为四代：机器语言、汇编语言、过程语言、面向对象语言。1970 年代时，基于过程的结构化编程方法占据主导地方。它认为程序只应该包含顺序、分支、循环等三种结构，GOTO 跳转大大地不好，应该禁止。而 Knuth 认为只要使用得当，GOTO 没什么不好的。

人们在编程时为了使程序清晰，常常在代码间插入注释。Knuth 认为这不够人性化，他主张按照程序员的思维逻辑，在注释间插入代码，这就是文学编程 (literate programming) 。

包子曰：于我心有戚戚焉。包老师写的东西往往注释连篇累牍，八卦亦真亦幻，正文不知所云。突然有一天，正文不见了，剩下的都是八卦，老包便已登堂入室，或可荣膺 Nobel Zhuangbility Prize。

1981 年，Knuth 提出了 WEB 编程系统，它把程序分成编程和文档两部分，编程部分就是 Pascal，文档部分则是 TEX。1987 年普林斯顿的 Silvio Levy (1959–) [20] 提出 CWEB，把编程部分换成 C 语言。

格式

TEX 是一种语言也是一个排版引擎 (engine) ，引擎的基本功能就是把字排成行，把行排成页，涉及到断字、断行、分页等算法。基本的 TEX 系统只有 300 多个元命令 (primitive) ，十分精悍，但是很难读懂，只适于非正常

[20] 1979 年巴西理论数学与应用数学研究所 (IMPA) 硕士。普林斯顿数学系 1985 年博士，1986 年博士后。1988 年加入明尼苏达大学几何中心，1995 年跳槽到陈省身 (1911–2004) 创立的数学科学研究所 (Mathematical Sciences Research Institute, MSRI) 。

人类。所以 Knuth 提供了一种格式 (format，宏命令的集合) 对 TEX 进行了封装，这就是 Plain TEX，包含 600 多个宏命令，然而它还是不够高级。

1980 年代初期，斯坦福研究所 (Stanford Research Institute, SRI) 的 Leslie Lamport (1941–) [21] 开发了一种新的格式，也就是 LATEX。1992 年 LATEX 2.09 发布后，Lamport 退居二线，之后的开发活动由 Frank Mittelbach [22] 等人接管。他们发布的最后版本是 1994 年的 LATEX 2_ε，LATEX 3 的开发也在进行中，只是正式版看起来遥遥无期。

宏包

LATEX 出现之后，在它的基础上出现了很多宏包 (package) 。起初，美国数学学会 (American Mathematical Society，AMS) 看着 TEX 是好的，就派 Michael D. Spivak (1940–) [23] 开发基于 Plain TEX 的宏包 $\mathcal{AMS}$-TEX，它的开发进行了两年 (1983–1985) 。后来与时俱进的 AMS 又看着 LATEX 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 [24] 刚刚搞了个字体系统 new font selection scheme for LATEX(NFSS) ，AMS 看着还不错，就拜托他们把 AMSFonts 加入 LATEX，继而在 1989 年请他们开发 $\mathcal{AMS}$-LATEX。次年 $\mathcal{AMS}$-LATEX 正式发布，之后它被整合为 $\mathcal{AMS}$ 宏包。

驱动

Knuth 最初设计的 TEX 只能用于施乐图形打印机 (Xerox graphic printer, XGP) ，这台打印机本身还需要一台 PDP-6 为它服务。1979 年，David R. Fuchs [25] 提出把 TEX 的输出改为设备无关的格式，也就是 DVI。

DVI 和其他页面描述语言的主要区别是，它不能嵌入字体和图形。所以它只能称作准页面描述语言，用户需要用驱动程序 (driver) 把它转换为其它格式，比如 PostScript 或 PDF 等。

---

[21] 1960 年 MIT 数学学士，布兰迪斯大学 (Brandeis University) 数学系 1963 年硕士，1972 年博士。1970 年加入麻省计算机同伙公司 (Massachusetts Computer Associates, MCA) ，1977 年跳槽到 SRI，1985 年跳到 DEC，2001 年跳到微软。

[22] 毕业于约翰内斯·古腾堡大学 (Johannes-Gutenberg University) 。1989 年加入电子数据系统公司 (Electronic Data Systems, EDS) 。

[23] 1964 年普林斯顿数学博士。

[24] LATEX3 的开发者之一。

[25] 普林斯顿毕业后，1978 年进入斯坦福攻读电脑博士。他不是 Knuth 的学生，但是完成过一些 TEX 的开发任务。在 Adobe 工作过一段时间，后混入娱乐圈，电影 *Red Diaper Baby*和 *Haiku Tunnel*的制片人。

1985 年斯坦福电脑博士生 Tomas Rokicki [26] 在 Funch 和 Knuth 的帮助下开发了 `dvips`，它把 DVI 转为 PostScript。后来维护它的是 TeX 用户组 (TeX Users Group, TUG) 的 Karl Berry [27] 等人。

1996 年，俄罗斯高能物理研究所 (Institute for High Energy Physics) 的 Sergey Lesenko发布了 `dvipdf`，它把 DVI 转为 PDF。后来凯特林大学 (Kettering University) 的 Mark A. Wicks开发的 `dvipdfm` 可以更好地处理字体、图形，它在 2001 年之后基本停止开发。`dvipdfm` 只能处理单字节字符，后来日本的平田俊作 (Shunsaku Hirata) 和韩国的赵珍焕 (Jin-Hwan Cho, 1968–) [28] 分别把它扩展为 `dvipdfm-jpn` 和 `dvipdfm-kor`，2002 年两者被合并为 `dvipdfmx`。它在字体、编码、图形方面都更优秀。

革命

1990 年代初，Knuth 和 Jiří Zlatuška (1957–) [29]、Philip Taylor (1947–) [30] 等探讨过怎样改进扩展 TEX，但是 Knuth 坚持只有他才能改动 TEX。1992 年这两位就纠集人马，在一位匿名赞助者的支持下搞了个新项目 new typesetting system (NTS)，企图改朝换代。他们看不上文学编程，就派 Karel Skoupý 把 TEX 逆向工程，还想换用 Java。

这次农民起义运动由于其历史局限性，缺乏马克思主义的世界观、方法论和一般原理的引导，同时遭到统治阶级的镇压，最终失败了。但是它留下了革命的火种：ε-TEX，把 TEX 支持的字符从 256 个扩充到 32768 个。

1994 年，Zlatuška 让他的学生 Hàn Thế Thành (1972–) [31] 试试改进 TEX 直接生成 PDF，这就是后来的 pdfTeX。1996 年马萨里克大学授予 Knuth 荣誉数学博士，他亲切地接见了 Hàn，并拍着他的小脑袋说，TEX 要从娃娃抓起。

[26] 1985 年毕业于得克萨斯农机大学后，到斯坦福念博士。博士毕业后加入惠普，1999 年离开创业。

[27] 1980 年代中期就读于马萨诸塞大学。毕业后跟着 Stallman 在 FSF 混，之后混迹于因特里弗 (Interleaf)、哈佛、直觉公司 (Intuit) 等地，2003 年起任 TUG 总统。

[28] 1999 年数学博士，日本大阪市立大学博士后。2001 年韩国高等研究所研究员，2004 年水原大学讲师。

[29] 捷克马萨里克大学 (Masaryk University) 电脑教授。

[30] 16 岁起在英国邮局总局 (General Post Office, GPO) 当技工，业余就读于几家技校。1970 年跳槽到莫林斯烟草机械 (Molins Tobacco Machinery)，1972 年跳到伦敦大学，1997 年起任该校皇家霍洛威学院 (Royal Holloway) 网管。曾任英国 TeX 用户组主席。

[31] 1991 年进入马萨里克大学信息系，1996 年硕士，2001 年博士。毕业后任教于胡志明市师范大学 (Ho Chi Minh City University of Pedagogy)。2006 年移居德国，现任河谷科技 (River Valley Technologies) 顾问。包老师猜他的名字应该翻译成韩世城。

Hàn 回越南期间，上网不便，其他一些人参与了 pdfTeX 的开发。2009 年底开发组决定除了修正漏洞外基本停止 pdfTeX 的开发，因为他们要转向一种新的引擎，LuaTEX。目前 LuaTEX 还处于测试阶段。

TEX 和 pdfTeX 的字体配置过程都很繁琐。2004 年初 Jonathan Kew [32] 发布的 XETEX 支持 Unicode 字符集和 AAT 字体，但是只能用于 Mac OS X。2005 年加入了对 OpenType 字体的支持，2006 年移植到 Linux 和 Microsoft Windows，2007 年被纳入 TeX Live 2007 和 MikTeX 2.7 发行版。

XETEX 工作时有两个步骤，第一步输出中间文件，Extended DVI (xdv)，第二步用驱动把 xdv 转为 PDF。缺省方式是两步一起执行，直接输出 PDF，xdv 只在内存中露过一小脸儿。用户也可以要求只执行第一步，保存 xdv。XETEX 现有两个驱动，一个是通用的 `xdvipdfmx`，看名字就知道它是 `dvipdfmx` 的亲戚；另一个是 Mac OS X 上的 `xdv2pdf`。

1990 年代初 Hans Hagen [33] 等开发的 ConTEXt 是在格式方面的一员革命小将。它一方面支持先进的语言、字体和排版技术，另一方面可以任意选择 TEX, pdfTeX, LuaTEX 等引擎。除了功能上的差异，它们在以下几方面也有所不同，

- 架构，LATEX 内核小，很多功能通过宏包来实现；ConTEXt 内核大，事无巨细都亲力亲为。
- 宏包，LATEX 谁都可以参与设计宏包，ConTEXt 则比较封闭。LATEX 的宏包多了，它们之间会有冲突；ConTEXt 没这个问题。
- 运行，LATEX 占用内存小，速度快；ConTEXt 占用内存大，速度慢。

1.1.4 小结

包老师语重心长地总结道，数字排版有四个重要环节：标记语言、页面描述语言、光栅图像处理器、输出设备。TEX 是最精确、最高级的面向专业排版的标记语言。TEX 家族可以划分为四个层次：引擎、格式、宏包、驱动。包老师通常选择 XETEX 引擎和 LATEX 格式。

1.2 优点缺点

通过上节内容我们已经知道，TEX 相对于其他标记语言有较大优势，但是在桌面印刷领域还有一种不可忽视的类别，所见即所得 (WYSIWYG) 系

[32] 毕业于剑桥大学。1985 年加入一家从事语言教育的非盈利组织 SIL。

[33] 1986 年创办普瑞格玛 (Pragma) 。NTS, pdfTeX, LuaTEX 等项目的参与者。

统，比如微软的 Word。其实 Word 也有自己的域代码 (field code) ，只是一般用户不太了解。

一般而言，TEX 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 TEX 软件都是免费的，虽然也有一些商业软件。

相应地，TEX 由于其工作流程，设计原则，资源的缺乏，以及历史局限性等原因也存在一些缺陷：

- 语法不如 HTML、XML 严谨、清晰。
- 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。
- 排版样式比较统一，但因而缺乏灵活性。
- 相对于商业软件，用户支持不够好，文档不完善。

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 TEX 当前的水平，工作量之大不是单枪匹马所能完成 [34]。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。”[35]

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

初学者面对上述那些引擎、格式、宏包、驱动等概念可能手足无措，所幸是有好事者把这些东东连同一些实用程序 (utilities) ，遵照 TEX 的规范打包集成在一起，形成一个发行版 (distribution) 或者说实现 (implementation) 。

与此类似的例子有 Java 和 Linux，比如 Sun、IBM、BEA [36]等公司都有

[34] TEX 也不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。正所谓一将功成万骨枯。

[35] 他果然于次年加入微软。

[36] 1995 年 Sun 的几位员工另立门户成立了 BEA，2008 年被 Oracle收购，2009 年 Oracle 又收购了 Sun。包子曰：天下大势，分久必合。

自己的 Java 虚拟机，它们都被称作 Java 的实现；而 Linux 有 Red Hat/Fedora、Ubuntu、SUSE 等发行版。

发行版的基本作用是提供 TEX 后台处理机制和命令行程序，用户还需要前台编辑器来编辑源文件。有的发行版也会附带一个编辑器，但是未必符合用户的口味。常用的 TEX 发行版和编辑器见 表 1.2。在使用 TEX 的过程中可能还需要其它一些软件，包老师会在相关章节中分别介绍。

表 1.2: TEX 发行版与编辑器

操作系统	发行版	编辑器
通用	TeX Live	TeXworks
Windows	MikTeX	TeXstudio
Mac OS	MacTeX	TeXShop

成熟稳重历经沧桑的用户更喜欢通用的编辑器，比如 Emacs 和 Vim，包老师喜欢轻量级的 PSPad。专用编辑器中，TeXstudio 和 TeXnicCenter 功能较强。Eclipse 有一个插件 TeXlipse，工作流程管理做得较好；但是 Eclipse 的自动换行有毛病，写程序还行，写文章就不合适了。

1.4 学习方法

限于篇幅和水平，本文只能介绍一些皮毛。最流行的英文入门资料是 Tobias Oetiker (1969–) [37] 的 *lshort*[1]，若想全面深入地了解 LATEX，可以拜读 Mittelbach 的 *LATEXCompanion*[2]。

Comprehensive TeX Archive Network (CTAN) 和TUG都提供了丰富的资源。常用宏包的简介见TeX Catalogue。常见问题可以参考英国 TeX 用户组 (UK TUG) 的TeX Frequently Asked Questions。

中文资料可参考李果正的《大家来学 LATEX》[3]，*lshort*有吴凌云 (1975–) [38] 等的中文译本 [39]。中文 TEX 论坛有水木清华 BBS TeX 版、CTeX 论坛。中文问题可参考 CTeX FAQ[4]。

在学习过程中，我们要先宏观后微观：先对体系、结构、框架等有所了解，慢慢再掌握使用方法，细节处理更要靠经验的积累。同时还要勤于思考 what、why、how 等问题。

[37] 奥腾大学 (Kantonsschule Olten) 学士，苏黎世联邦理工学院 (Swiss Federal Institute of Technology Zurich) 电子硕士，1995 年留校工作。

[38] 1997 年武汉大学应用数学学士，2002 年中科院运筹学博士，香港科技大学、中科院、康奈尔博士后，2007 年中科院数学与系统科学研究院副研究员。

[39] 就在英文版的隔壁，更新得慢一点。

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

参考文献

[1] Tobias Oetiker. *A (Not So) Short Introduction to LaTeX2e*, 2009. URL http://www.ctan.org/tex-archive/info/lshort/.

[2] Frank Mittelbach, Michel Goossens, Johannes Braams, David P. Carlisle, and Chris Rowley. *The LaTeX Companion (Tools and Techniques for Computer Typesetting)*. Addison-Wesley, 2nd edition, 2004. URL http://www.amazon.com/dp/0201362996/.

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

[4] 中国 TeX 用户组. *CTeX* 常见问题集, 2005. URL http://www.ctan.org/tex-archive/info/ctex-faq/.

第二章　入门

2.1　Hello, World!

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

例 2.1: Hello, Wolrd!

在编辑器中将 例 2.1 中的代码保存为 `hello_world.tex`，这就是一份最简单的 LaTeX 源文件。然后我们可以用 `xelatex` 程序编译源文件生成 PDF，它知道输入的是 LaTeX 源文件，所以这里的 `.tex` 后缀可以省略。以后类似情况都用括号标出，不再特意声明。

```
xelatex hello_world(.tex)
```

如果系统显示类似下面的错误信息，请检查源文件中是否有拼写错误。`.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.pdf (1 page).
Transcript written on hello_world.log.
```

TEX 系统针对不同格式和引擎的组合，提供了一系列的命令行程序，完成不同的编译和转换功能。比如源文件是 plain TEX 格式时，可以分别用 `tex`、`pdftex`、`xetex` 程序调用 TEX, pdfTeX, XƎTEX 引擎；源文件是 LATEX 格式时，相应的程序则是 `latex`，`pdflatex`，`xelatex` [1]。选择编译和转换程序时可以参考 图 2.1，一般有直接方法可用时，不必非要排个转折亲使用间接方法。

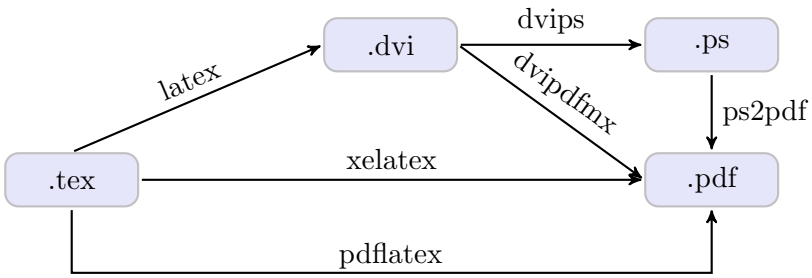


图 2.1: 编译和格式转换

2.2 语法和结构

2.2.1 语法

LATEX 源文件的语句可以分为三种：命令 (command) 、数据和注释 (comment) 。命令又分为普通命令和环境 (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，用于多行内容的场合。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。

比如在 例 2.1 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。

2.2.2 物理结构

LATEX 文档的结构可以分为物理结构和逻辑结构。前者指的是源文件的组织形式，包括序言 (preamble) 和正文两部分；后者则是最终输出文档的结

[1] 现在的发行包大多以 pdfTeX 为缺省引擎，所以 `latex` 命令缺省调用的其实是 pdfTeX，而不是 TEX。

构，包括标题、目录、章节等。这里只简要介绍一些基本概念，在第七章还会展开详谈。

序言用来完成一些设置，比如指定文档类型，引入宏包，定义命令、环境等；文档的实际内容则放在正文部分。它们的基本用法如下：

```
\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`，它们的基本选项见 表 2.1。

注意

表 2.1: 文档类常用选项

10pt, 11pt, 12pt	正文字号，缺省 10pt。LaTeX 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 letterpaper。
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 缺省用单面，book 缺省用双面。
landscape	横向打印，缺省是纵向。
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。

LaTeX 核心只提供基本功能，很多功能要通过宏包来实现。其他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`。

2.2.3 逻辑结构

一份文档的开头通常有标题、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。文档比较长时我们还可以使用目录。

标题、作者、日期等命令用法如下，注意 `\maketitle` 命令要放在最后。

```
\title{LaTeX Notes}
\author{Alpha Huang}
\date{\today}
\maketitle
```

article 和 report 可以有摘要，book 里没有。摘要环境用法如下：

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

LaTeX 提供了七种层次结构命令，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 则支持所有层次。

```
\part{...}            %Level -1
\chapter{...}         %Level 0
\section{...}         %Level 1
\subsection{...}      %Level 2
\subsubsection{...}   %Level 3
\paragraph{...}       %Level 4
\subparagraph{...}    %Level 5
```

例 2.2: 层次结构

我们可以用 `\tableofcontents` 命令来生成目录。系统会自动设定目录包含的章节层次，用户也可以显式指定目录层次深度。比如下面的命令指定目录深度为 2，也就是只显示 subsection 及以上层次的目录。注意设定目录深度命令要放在列目录命令的前面。

```
\setcounter{tocdepth}{2} %设定目录深度
\tableofcontents         %列出目录
```

初次使用目录，或章节图表等层次结构发生变化时，都需要执行两遍编译命令才能获得正确结果。LaTeX 之所以设计成这样是因为当时的电脑内存容量有限。

如果我们不想让某些层次的标题出现在目录里，则可以给 例 2.2 中的命令加上星号。

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

类似地，我们也可以用下面的命令生成插图和表格目录，插图和表格功能将在后面相关章节中介绍。这两种目录也都需要编译两遍。

```
\listoffigures
\listoftables
```

2.3 文字

文档的内容可以分为文本模式和数学模式。前者是缺省工作方式；要输入数学内容则需要特殊命令或环境。本章只涉及文本模式，第四章会介绍数学模式。

2.3.1 字符输入

文档中可以输入的文字符号大致可以分为：普通字符、控制符、特殊符号、预定义字符串、注音符号等。

普通字符可以直接输入，而有些字符 (例如 # $ % ^ & _ { } ~ 等) 被用作特殊的控制符，输入时多数需要在前面加个 \。而 \ 本身则要用 \textbackslash 命令来输入，因为 \\ 被用作换行指令[2]。

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

表 2.2 给出一些特殊符号和预定义字符串的输入方法；其中 \XeTeX、\XeLaTeX 命令需要 metalogo 宏包，\MF、\MP 命令需要 mflogo 宏包，\AmS 命令需要 texnames 宏包。表 2.3 列出一些注音符号。更多的符号见 Scott Pakin [3] 的符号列表[1]。

LaTeX 中有短划线 (hypen) 、中划线 (en-dash) 和长划线 (em-dash) 。短划线又称连字符，用来连接单词；中划线用来连接数字，可以通过重复两次短划线得到；长划线类似于中文的破折号，重复三次短划线。为了便于比较，这里也给出数学模式的减号。

[2]为什么不用 C 语言的 \n 呢，也许因为 TeX 的编程语言是 Pascal。

[3]UIUC计算机系 1995 年硕士，2001 年博士。现供职于洛斯阿莫斯国家实验室 (Los Alamos National Laboratory) 。

表 2.2: 特殊符号和预定义字符串

特殊符号		预定义字符串	
©	\textcopyright	January 17, 2012	\today
®	\textregistered	TEX	\TeX
™	\texttrademark	LATEX	\LaTeX
¥	\textyen	LATEX 2ε	\LaTeXe
€	\texteuro	XELATEX	\XeTeX
£	\pounds	XELATEX	\XeLaTeX
…	\dots	METAFONT	\MF
¶	\P	METAPOST	\MP
§	\S	AMS	\AmS

表 2.3: 注音符号

Ȧ	\.{A}	Ä	\"{A}	Ā	\={A}	À	\`{A}
Á	\'{A}	Â	\^{A}	Ă	\v{A}	Ã	\~{A}
Ạ	\d{A}	A̲	\b{A}	Å	\r{A}	Ⓐ	\textcircled{A}

2.3.2 字体样式和大小

拉丁文字体主要有三大类：衬线字体 (roman, serif) 、无衬线字体 (sans serif) [4]和等宽字体 (monospace, typewriter) 。衬线字体笔画的边缘部分有些修饰，类似于中文的宋体、仿宋、楷体、魏体等。无衬线字体的笔画则是平滑的，类似于中文的黑体。

字体还可以有粗体 (bold) 、斜体 (italic) 、伪斜体 (oblique, slanted) 、小型大写字母 (small caps) 等修饰效果。TEX 提供了介于正常字体和粗体之间的半粗体 (medium weight) 。斜体通常对原字体进行了重新设计，它修饰精细，多用于衬线字体；伪斜体基本上是把原字体倾斜，多用于无衬线字体，一般伪斜体看起来比斜体要宽一些。小型大写字母的形状和大写字母相同但尺寸较小，一般高度和小写字母相似。

每种字体样式包含很多种具体的字体，第三章会有关于字体更详细的介绍，包括中文字体的设置。字体常用样式命令见 表 2.4。

LATEX 的字体强调命令 `\emph` 在不同字体环境中有不同的效果。如果周围文字是正体，它就是斜体；反之它就是正体。`\underline` 命令可以给字体加下划线，但是它不能正确断字。`ulem` 宏包改进了断字，还增加了波浪线和删除线等命令。只是 `ulem` 把 `\emph` 重定义成了下划线，我们可以在引用宏包时可以加个选项改回去：`\usepackage[normalem]{ulem}`。

[4]Sans 这个词来源于法语，就是“没有”的意思。

```
computer-aided\\
1840--2010\\
to be---or not to be\\
$1-1=0$
```

computer-aided
1840–2010
to be—or not to be
$1-1=0$

例 2.3: 划线和减号

表 2.4: 字体样式

roman	`\textrm{...}`	**bold face**	`\textbf{...}`
sans serif	`\textsf{...}`	medium weight	`\textmd{...}`
`typewriter`	`\texttt{...}`	*italic*	`\textit{...}`
SMALL CAPS	`\textsc{...}`	slanted	`\textsl{...}`

LaTeX 会根据正文的字体大小来调整标题、章节、上下标、脚注等的字号。我们也可以用 表 2.5 中的命令来设置字体相对尺寸，比如正文的字号是 10pt、11pt、12pt 时，tiny 的字号就分别是 5pt、6pt、6pt。

表 2.5: 字体相对尺寸

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

2.3.3 换行、换页和断字

通常 LaTeX 会自动换行，我们也可以用 `\\` 或 `\newline` 命令来强制换行；用 `\newpage` 命令来强制换页。

LaTeX 也会自动断字 (hyphenate)，使得每一行的字间距分布均匀。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，

```
\emph{emphasis}\\
\uline{underline}\\
\uwave{waveline}\\
\sout{strike-out}
```

emphasis
underline
waveline
~~strike-out~~

例 2.4: 字体强调和下划线

而 blar-blar-blar 可以在 -处断开。

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

2.4 长度

为了精确排版，人们需要控制排版对象的尺寸和位置。LaTeX 中常用长度单位见 表 2.6，其中 point 是个传统印刷业采用的单位，而 big point 是 Adobe 推出 PostScript 时定义的新单位。em 是个相对单位，比如当前字体是 11pt 时，1em 就是 11pt；ex 和 mu 也是相对单位。

表 2.6: 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

LaTeX 为排版对象的尺寸和位置定义了一系列宏变量，以便在排版时重用。我们可以用下面的方法来改变宏变量的值或定义新的宏变量，

```
\setlength{变量名}    %设置变量的值
\addtolength{变量名}  %增加变量的值
\newlength{变量名}    %定义新变量
```

2.5 对齐和间距

2.5.1 段落对齐

LaTeX 中的段落缺省两端对齐 (fully justified) ，下面的三个环境可以让段落分别居左、居右或居中对齐。另有三个命令 (`\raggedright`, `\centering`, `\raggedleft`) 可以完成同样功能。

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

居左
段落

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

居右
段落

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

居中
段落

例 2.5: 段落对齐方式

2.5.2 缩进和段间距

LaTeX 正文中第一个段落缺省不缩进首行，我们可以用 identfirst 宏包使得第一段也缩进首行。段落首行缩进的距离可以用 \parindent 变量来控制，段落之间的距离可以用 \parskip 变量来控制。

```
\usepackage{identfirst}
...
\setlength{\parindent}{2em}
\addtolength{\parskip}{3pt}
```

2.5.3 行间距

行间距是段落中相邻两行基线之间的距离，LaTeX 缺省使用单倍行距。我们可以用 \linespread 命令来控制行距。

```
\linespread{1.3} %一倍半行距
\linespread{1.6} %双倍行距
```

\linespread 命令不仅会改变正文行距，同时也把目录、脚注、图表标题等的行距给改了。如果只想改正文行距，可以使用 setspace 宏包的行距命令（见 例 2.6）。

上述行距命令对全文的行距都会产生影响，setspace 宏包还提供了 singlespacing，onehalfspacing，doublespacing，spacing 等环境，可以用来设置局部文字的行距（见 例 2.7）。

```
\usepackage{setspace}
...
\singlespacing     %单倍行距
\onehalfspacing    %一倍半行距
\doublespacing     %双倍行距
\setstretch{1.25}  %任意行距
```

例 2.6: 行距命令

```
\begin{doublespacing}
double\\spacing
\end{doublespacing}
```

double
spacing

```
\begin{spacing}{1.25}
any\\spacing
\end{spacing}
```

any
spacing

例 2.7: 行距环境

2.6 特殊段落

2.6.1 摘录

LaTeX 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。

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

引文两端
都缩进。

```
\begin{quotation}
引文两端缩进，首行增加缩进。
\end{quotation}
```

引文两端缩进，首行增加缩进。

```
\begin{verse}
引文两端缩进，第二行起增加缩进。
\end{verse}
```

引文两端缩进，第二行
起增加缩进。

例 2.8: 摘录环境

2.6.2 原文打印

文档中的命令和源代码通常使用等宽字体，也就是原文打印。正文间插入少量等宽文字可以使用 `\verb` 命令；大段原文打印用 `verbatim` 环境比较方便，它的带星号版本可以标出空格。

```
\verb|command| 行间命令
\begin{verbatim}
printf("Hello, world!");
\end{verbatim}
\begin{verbatim*}
printf("Hello, world!");
\end{verbatim*}
```

command 行间命令

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

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

例 2.9: 原文打印

Timothy van Zandt [5] 等人的 `fancyvrb` 宏包[2] 和 Brooks Moses [6] 等人的 `listings` 宏包[3] 提供了更多的原文打印功能，比如行号、边框、背景、语法着色等。

2.6.3 脚注

脚注可以使用 `\footnote` 命令；如要改变脚注编号形式，可以使用以下命令。在 例 2.10 中，`footnote` 是一个计数器 (counter)；计数器有五种显示格式 (见 表 2.7)，重定义 `\thefootnote` 宏时可任选。

```
正文\footnote{脚注}
```

正文[a]

[a] 脚注

```
\renewcommand{\thefootnote}{\roman{footnote}} %i, ii, iii
```

例 2.10: 脚注

以后我们还会遇到其他一些计数器，都可以用重定义 `\thecounter` 的方法改变它们的显示格式。

原始的脚注命令不能包含原文打印命令或环境，我们可以用 `\texttt` 命令来输入等宽字体，或者用 `fancyvrb` 宏包的 `\VerbatimFootnotes` 命令

[5] 宾州大学经济系 1985 年学士，1989 年博士。1990 年加入普林斯顿，1998 年跳槽到欧洲工商管理学院。

[6] 1997 年弗吉尼亚理工学院机械学士，2007 年斯坦福机械博士，毕业后加入一家软件公司源码巫师 (CodeSourcery)。

表 2.7: 计数器显示格式

阿拉伯数字	\arabic{counter}	1, 2, 3…
小写英文字母	\alph{counter}	a, b, c…
小写英文字母	\Alph{counter}	A, B, C…
小写罗马数字	\roman{counter}	i, ii, iii…
大写罗马数字	\Roman{counter}	I, II, III…

重定义 \footnote 命令。

2.6.4 边注

边注可以使用 \marginpar 命令。单面排版时，边注缺省排在页面右边空白处；双面排版时，边注在外侧，也就是左页的左边或右页的右边；双栏页面的边注排在最近的页边。如要切换边注的方向，可以使用 \reversemarginpar 和 \normalmarginpar 命令。

\marginpar 命令使用浮动体 (float) [7] 来生成边注，所以不能在其他浮动体或脚注内嵌套。marginnote 宏包的 \marginnote 命令不使用浮动体，因而没有这个缺陷。

正常边注

```
\marginnote{正常边注}
\reversemarginpar
\marginnote{反向边注}
\normalmarginpar
```

反向边注

例 2.11: 边注

2.6.5 注释

前面提到可以用百分号来标明注释，但是对于大段文字的注释，百分号就显得比较繁琐 [8]。这时我们可以使用 verbatim 宏包的 comment 环境。

```
\begin{comment}
...
\end{comment}
```

[7] 浮动体详见第五章插图和第六章表格。

[8] 有的编辑器提供切换多行百分号的功能，比如 TeXNicCenter。

2.7 列表

2.7.1 基本列表

LaTeX 有三种基本列表环境：无序列表、有序列表、描述列表。这些列表可以单独使用，也可以互相嵌套。

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

- C++
- Java
- HTML

(a) 无序列表

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

1. C++
2. Java
3. HTML

(b) 有序列表

```
\begin{description}
    \item[C++] 编程语言
    \item[Java] 编程语言
    \item[HTML] 标记语言
\end{description}
```

C++ 编程语言

Java 编程语言

HTML 标记语言

(c) 描述列表

例 2.12: 基本列表

2.7.2 其他列表

上述列表的缺省行间距较大，如要节省空间，可以考虑 Bernd Schandl [9] 的 `paralist` 宏包，它提供了一系列压缩列表和行间列表环境。

[9] 1997 年凯撒斯劳腾工业大学 (Kaiserslautern University of Technology) 数学硕士，1999 年克莱姆森大学数学博士。

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

- C++
- Java
- HTML

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

1. C++
2. Java
3. HTML

```
\begin{compactdesc}
    \item[C++] 编程语言,
    \item[Java] 编程语言,
    \item[HTML] 标记语言。
\end{compactdesc}
```

C++ 编程语言,
Java 编程语言,
HTML 标记语言。

例 2.13: 压缩列表

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

• C++ • Java • HTML

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

1. C++ 2. Java 3. HTML

```
\begin{inparadesc}
    \item[C++] 编程语言,
    \item[Java] 编程语言,
    \item[HTML] 标记语言。
\end{inparadesc}
```

C++ 编程语言, **Java** 编程语言,
HTML 标记语言。

例 2.14: 行间列表

2.7.3 定制列表

如要改变无序列表的列表符号和有序列表的编号形式，可以使用以下代码，其效果见 例 2.15。

```
\renewcommand{\labelitemi}{-}
\renewcommand{\theenumi}{\alph{enumi}}
```

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

- C++
- Java
- HTML

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

a. C++
b. Java
c. HTML

例 2.15: 定制列表

2.8 盒子

LaTeX 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 初级盒子

最简单的盒子命令是 `\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}
```

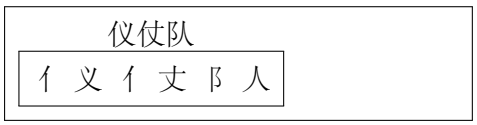
010 6278 5001 010 6278 5001

例 2.16: `mbox`和 `fbox`

2.8.2 中级盒子

稍复杂的 `\makebox` 和 `\framebox` 命令提供了宽度和对齐方式控制的选项。其对齐方式有居中 (缺省) 、居左、居右和分散对齐，分别用 `c`、`l`、`r`、`s` 来表示。

```
%语法：[宽度][对齐方式]{内容}
\makebox[100pt][c]{仪仗队}
\framebox[100pt][s]{仪仗队}
```



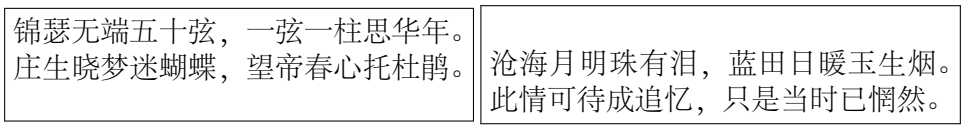
例 2.17: makebox和 framebox

2.8.3 高级盒子

大一些的对象比如整个段落可以用 \parbox 命令或 minipage 环境，两者语法类似，有宽度、高度、外部对齐、内部对齐等选项。这里的外部对齐是指该盒子与周围对象的纵向关系，有三种方式：居顶、居中和居底对齐，分别用 t、c、b 来表示。内部对齐是指该盒子内部内容的纵向排列方式，也是同样三种。

语法：[外部对齐][高度][内部对齐]{宽度}{内容}

```
\fbox{%
    \parbox[c][36pt][t]{170pt}{
    锦瑟无端五十弦，一弦一柱思华年。庄生晓梦迷蝴蝶，望帝春心托杜鹃。
    }%
}
\hfill
\fbox{%
    \begin{minipage}[c][36pt][b]{170pt}
    沧海月明珠有泪，蓝田日暖玉生烟。此情可待成追忆，只是当时已惘然。
    \end{minipage}%
}
```



例 2.18: parbox 和 minipage

2.9 交叉引用

在 LaTeX 中我们可以任意设置标签，然后引用标签前面最近一个对象 (章、节、图、表等) 的编号或者页码，这就是交叉引用 (cross reference) 。例 2.19 中的 marker 是一个标签名，它在全文中须保持唯一。

文档中新增标签后后，第一次编译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标签的地方，第二次编译才能得到正确结果。

```
一个标签\label{marker}
...
第\pageref{marker}页\ref{marker}节
```

一个标签
...
第 30 页 2.9 节

例 2.19: 交叉引用

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

参考文献

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

[2] Timothy van Zandt. *Fancy Verbatims in LaTeX*, 2008. URL http://www.ctan.org/tex-archive/macros/latex/contrib/fancyvrb/.

[3] Carsten Heinz and Brooks Moses. *The listings Package*, 2007. URL http://www.ctan.org/tex-archive/macros/latex/contrib/listings/.

广告位招租

第三章　字体

英文中的 typeface 和 font 一般都被翻译为字体，传统印刷业通常使用 typeface，电脑字体通常使用 font；当然也有很多人混用这两个词。包老师认为电脑字体的诸多相关概念可以划分为三个层次：

1. 编码层，字符 (包括字母、数字、符号、控制码等) 的索引和编码，也就是字符集 (character set) 和字符编码 (character encoding) 。
2. 格式层，字形 (glyph) 的定义描述方法，以及字体的文件存储格式。
3. 显示层，字体的外在表现形式，比如字体的样式，或具体的字体。

看到这里好奇的读者可能会问，为什么 Microsoft Word 和其他所见即所得软件可以直接用下拉菜单设置字体，同样是标记语言的 HTML 也可以方便地设置字体，而 LaTeX 就要牵扯这么多概念？我们不得不承认，这是 LaTeX 的历史局限性之一。Knuth 当初设计 TeX 时，既没有 Unicode 字符集和 UTF-8 编码，也没有 TrueType 和 OpenType 字体格式。

包老师语重心长地说，道路是曲折的，前途是光明的。这不马上就要侃到 XeLaTeX 了，这些困难只是黎明前的黑暗。

3.1　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。所有字符在电脑内部都是用二进制表示的，字符集的二进制编码被称为字符编码，有时人们也会混用这两个术语。

1963 年 ANSI发布了基于电报码的 ASCII，这就是最早的字符编码，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，他们优先考虑英语是可以理解的。后来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) ，可以涵盖 $2^8 = 256$ 个字符。

于是 ISO 在 1980 年代中期推出了 ISO 8859，256 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，也就是代码页 (code page)，比如 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，比如 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 94×94 的表，包括 7445 个字符。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 中没有朱镕基的“镕”字，于是它在 1993 年被扩展为 GBK，包含 21886 个字符。GBK 不是正式标准。2000 年发布的 GB18030 包含 70244 个字符，采用四字节编码。GB18030 之前还出现过一个 GB13000，但是没有形成气候。

1990 年 ISO 推出了通用字符集 (universal character set，UCS)，即 ISO 10646，意图一统江湖。它的容量是一百多万个字符，目前实际使用的有十万个左右。UCS 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，于是双方开始合并成果，携手奔小康。Unicode 从 2.0 版开始采用与 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。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。

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。

3.2 字体格式

3.2.1 点阵和矢量字体

电脑字体的数据格式可以分为三大类：点阵 (bitmap) 字体、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。

3.2.2 常见字体格式

当前常见的轮廓字体格式有：Type 1、TrueType、OpenType。

1984 年 Adobe 推出 PostScript 时，同时支持两种字体格式：Type 1 和 Type 3，它们都采用三次贝塞尔曲线。Type 1 支持微调，它使用一个简化的 PostScript 子集；Type 3 不支持微调，但它可以使用全部 PostScript 功能，因此既可包含轮廓字体也可包含点阵字体信息。

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 开始研究 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，它可以被认为是 Type 1 和 TrueType 的超集，既可使用二次曲线，也可使用三次曲线。它比起 TrueType 和 AAT 的优势还有：平台独立、开放、易于开发，并且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 把字体的尺寸 (metrics) 信息和字形 (glyph) 信息分别存储。字体尺寸文件有 AFM (Adobe font metrics) 和 PFM (printer font metrics)，字形文件有 PFA (printer font ASCII) 和 PFB (printer font binary)。LaTeX 使用的尺寸格式是 TFM (TeX font metrics)。

TrueType 和 OpenType 则将字体数据都存在一个文件里，它们的文件后缀分别是是.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)。

这些字体格式按照技术先进性，从高到低依次为：OpenType、TrueType、Type 1、Type 3、PK，所以我们要优先选用 OpenType 和 TrueType。

PostScript 文件可以包含 Type 1 和 Type 3 字体，而 PDF 除了这两种还支持 TrueType 和 OpenType 字体。

3.2.3 合纵连横

当年 Adobe 推出 Type 1 和 Type 3 时，前者收费，后者是公开的自由规范。Type 1 专利许可费十分昂贵，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，它从贵族堕落为平民，因而流行开来。

1980 年代中后期，Adobe 的大部分盈利来自于 PostScript 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 PostScript 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。

3.3 常见字体

在 2.3.2 节中我们提到每种字体样式可以包含很多种具体的字体。为了方便读者，表 3.1 列出一些最常见的字体。TEX 的缺省字体是 Knuth 用 METAFONT生成的 Computer Modern；XETEX 的缺省字体是 1997 年 $\mathcal{AMS}$ 发布的 Latin Modern，它基于 Computer Modern，但是扩展了其字符集，其封装格式有 Type 1 和 OpenType。

表 3.1: 常见字体

操作系统	衬线字体	无衬线字体	等宽字体
Windows	Times New Roman Georgia Palatino Linotype	Tahoma Verdana Arial	Courier New Lucida Console Consolas
Mac OS	Times Georgia Times New Roman	Helvetica Lucida Grande Geneva	Monaco Courier Courier New

3.4 字体的应用

从理论上讲，任何电脑字体只要有 TFM，TEX 就可以使用它。然而早期的 TEX 只能使用 METAFONT生成的字体。直到LATEX 2$_\varepsilon$时代 NFSS 的出现后，Type 1 和 Type 3 才在 LATEX 中得到广泛应用。后起之秀 XETEX 则极大程度地简化了 TrueType 和 OpenType 的配置，而且它还支持 Unicode。

`latex`、`pdflatex`、`xelatex` 编译程序，`dvips` 和 `dvipdfmx` 驱动，DVI 浏览器等分别采用不同的字体技术路线。本文主推 XELATEX，对其他早期技术只作简要介绍，聊以忆苦思甜。

3.4.1 早期技术

latex 和 DVI

我们用 `latex` 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。

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 PostScript 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 PostScript 的 Type 3。因此含 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 PostScript，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[1] 第六章，此处不赘述。

dvipdfm(x) 和 pdflatex

`dvipdfm` 支持 PK 和 Type 1，它可以用一个 `t1fonts.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；只是其安装配置较繁琐，此处不赘述，实在有兴趣考古的读者可以参考 lnotes 第一版[2]。

pdflatex 支持 Type 1、TrueType，也在一定程度上支持 OpenType。它的配置也很繁琐，可以参考其用户手册[3] 第五章。

3.4.2 XeTeX

XeTeX 可以直接使用电脑系统字体，不再需要 TFM 文件。我们首先需要知道电脑上有哪些字体，XeTeX 用一个 XML 文件记录系统字体路径，MikTeX 用的是 `localfonts.conf`，TeXlive 用的是 `fonts.conf`。

我们设置字体时需要字体的引用名，它和字体的文件名是不同的概念。`fc-list` 程序可以用来获取字体引用名，比如下面命令生成的 `myfonts.txt` 文件就是一份字体引用名列表。

```
fc-list > myfonts.txt
```

带 XeTeX 的发行包首次安装时会自动扫描这些字体目录，生成字体的

缓存 (cache) 。每次系统安装了新字体时，我们需要手工运行字体缓存命令 `fc-cache`，生成新的缓存。

```
fc-cache -r
```

关于字体路径，`fc-list` 和 `fc-cache` 命令等的详细信息，可以参考 Michel Goossens (1951–) [1] 编辑的 *XeTeX Companion*[4]。

XeTeX 提供的字体命令比较原始、繁琐，Will Robertson (1981–) [2] 的 `fontspec` 宏包提供了较好的封装。XeLaTeX 下字体常用设置方法如下，详细信息可以参考 `fontspec` 的用户手册[5]。

```
\usepackage{fontspec}
\setmainfont[Mapping=tex-text]{Times New Roman}
\setsansfont[Mapping=tex-text]{Tahoma}
\setmonofont{Courier New}
```

例 3.1: XeLaTeX 字体设置

例 3.1 中的代码分别设置了衬线、无衬线和等宽字体样式对应的字体。在 2.3.1 节 例 2.3 中我们学过中划线和短划线可以用 `--` 和 `---` 来输入。TeX 中几个短划线是相连的，XeTeX 中缺省它们之间是有空隙的。例 3.1 用 `Mapping` 参数指示改回 TeX 的方式，即去掉短划线之间的空隙。

3.5 中文解决方案

LaTeX 对中文的支持主要有两种方法：张林波 [3] 的 CCT 和 Werner Lemberg (1968–) [4] 的 CJK 宏包。早期 CCT 比较流行，CJK 后来居上；新版 CCT 也可以和 CJK 配合使用。

支持简体中文的 LaTeX 发行版有吴凌云的 CTeX 和李树钧 [5] 的 ChinaTeX，

[1] 布鲁塞尔自由大学 (Free University of Brussels) 物理系 1972 年学士，1978 年博士。1979 年加入 CERN。合著 *LaTeX Companion*、*LaTeX Graphics Companion*、*LaTeX Web Companion*等书。曾任 TUG 总统。

[2] 阿德雷德大学 (University of Adelaide) 机械系博士生，80 后帅哥。

[3] 中科院数学与系统科学研究院研究员。

[4] 从维也纳音乐和表演艺术大学 (University of Music and Performing Arts, Vienna) 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。

[5] 西安交大电子系 1997 年学士，2003 年博士。2003 年香港城市大学博士后，2005 年香港理工大学博士后，2007 年哈根函授大学 (Distance University of Hagen) 研究员，2008 年康斯坦茨大学 (University of Konstanz) 研究员。

支持繁体中文的有吴聪敏 (1952–) [6]、吴聪慧 [7] 兄弟和翁鸿翎的 cwTeX，蔡奇伟 [8] 的 PUTeX。这两个繁体中文的发行版本人都不熟悉，前两个简体中文发行版都包含 MikTeX、CCT、CJK、WinEdt 等。

XeTeX 问世后，孙文昌 (1970–) [9] 推出了 `xeCJK` 宏包，用于排版中日韩等文字，包括字体选择，标点、文字间距调整等功能。它可以被认为是 CCT 和 CJK 在某种程度上的结合。

在 例 3.2 中引用宏包时，`CJKchecksingle` 参数防止段落最后一行只有一个汉字；`CJKnumber` 参数自动载入另一个宏包 `CJKnumber`，它提供的 `\CJKnumber` 命令可以把阿拉伯数字转换为中文数字。

`\setCJKmainfont` 命令设置了中文正文字体，它的两个参数 `BoldFont` 和 `ItalicFont` 分别设置了粗体、斜体样式对应的字体。`\setCJKsansfont`、`\setCJKmonofont` 命令分别设置了无衬线和等宽字体样式对应的字体。

```
\usepackage[CJKaddspaces,CJKchecksingle,CJKnumber]{xeCJK}
\setCJKmainfont[BoldFont={Adobe Heiti Std},
    ItalicFont={Adobe Kaiti Std}]{Adobe Song Std}
\setCJKsansfont{Adobe Heiti Std}
\setCJKmonofont{Adobe Fangsong Std}
\punctstyle{hangmobanjiao}
```

例 3.2: xeCJK

其实严格地讲中文字体并没有衬线、无衬线、等宽、斜体等概念，只是习惯上宋体用得最多，辅以黑体、仿宋、楷体，文档会显得疏落有致，不至于太沉闷。如果我们把 例 3.1 和 例 3.2 中的命令结合起来，就可以为中英文分别设置字体。

xeCJK 宏包的详细用法可参考其用户手册[6]。另外 Yin Dian 的 zhspacing 宏包也可以完成类似功能，具体用法可参考其文档[7]。

参考文献

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

[6]台湾大学电机学士，罗彻斯特大学经济学博士，台湾大学经济系教授。

[7]新泽西理工学院电脑硕士，嘉南药理科技大学医务管理系助理教授。

[8]犹他电脑博士，静宜大学资讯工程系副教授。

[9]南开大学数学系 1993 年学士，1998 年博士，毕业后留校作博士后，2000 年副教授，2002 年教授。

[2] 包太雷. *LaTeX Notes*, 2008. URL http://www.ctan.org/tex-archive/info/latex-notes-zh-cn/.

[3] Thé Thành Hàn, Sebastian Rahtz, Hans Hagen, Hartmut Henkel, Paweł Jackowski, and Martin Schröder. *The pdfTeX User Manual*, 2007. URL http://www.tug.org/applications/pdftex/.

[4] Michel Goossens. *The XeTeX Companion*, 2010. URL http://xml.web.cern.ch/XML/lgc2/xetexmain.pdf.

[5] Will Robertson. *The fontspec Package*, 2008. URL http://www.ctan.org/tex-archive/macros/xetex/latex/fontspec/.

[6] 孙文昌. *xeCJK 宏包*, 2009. URL http://www.ctan.org/tex-archive/macros/xetex/latex/xecjk/.

[7] Dian Yin. *Typesetting Chinese in XeTeX: zhspacing User's Manual*, 2007. URL http://code.google.com/p/zhspacing/.

广告位招租

第四章　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 $\mathcal{AMS}$-LaTeX 提供的数学功能，我们需要在文档的序言部分加载 `amsmath` 宏包，其详细用法可参阅其用户手册[1]。更全面的数学内容排版可参阅 George Grätzer [1] 的 *More Math into LaTeX* [2]。

```
\usepackage{amsmath}
```

4.1　数学模式

LaTeX 的数学模式有两种形式：行间 (inline) 模式和独立 (display) 模式。前者是指在正文中插入数学内容；后者独立排列，可以有或没有编号。简单数学公式的输入方法见 表 4.1。

行间公式和无编号独立公式都有多种输入方法，新手也许会看花了眼。懒人包老师的秘诀是用最短的：行间公式用 `$...$`，无编号独立公式用 `\[...\]`。建议不要用 `$$...$$`，因为它和 $\mathcal{AMS}$-LaTeX 有冲突。`amsmath` 版本的 `equation` 环境可以嵌入次环境 (见 4.4.1 节) 。

[1] 匈牙利厄特沃什·罗兰大学 (Eötvös Loránd University) 1960 年数学博士，现任加拿大曼尼托巴大学 (University of Manitoba) 教授。John von Neumann 的校友。

表 4.1: 简单数学公式的输入

	TEX 命令	LATEX 命令	LATEX 环境	amsmath 环境
行间公式	`$...$`	`\(...\)`	`math`	
无编号独立公式	`$$...$$`	`\[...\]`	`displaymath`	`equation*`
有编号独立公式			`equation`	`equation`

2.8 节提到的 `\fbox` 命令可以给文本内容加个方框，数学模式下也有个类似的命令 `\boxed`。

```
Einstein's $E=mc^2$
\[ E=mc^2 \]
\[ \boxed{E=mc^2} \]
\begin{equation}
    E=mc^2
\end{equation}
```

Einstein's $E = mc^2$

$$E = mc^2$$

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

$$E = mc^2 \tag{4.1}$$

例 4.1: 数学模式

4.2 基本元素

4.2.1 希腊字母

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 4.2 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 4.2: 希腊字母

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`		

4.2.2 上下标和根号

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。

```
\[ x_{ij}^2\quad \sqrt{x}\quad \sqrt[3]{x} \]
```

$$x_{ij}^2 \quad \sqrt{x} \quad \sqrt[3]{x}$$

例 4.2: 上下标和根号

4.2.3 分数

分数用 \frac 命令表示，它会根据环境自动调整字号，比如在行间公式中小一点，在独立公式中则大一点。我们可以人工设置分数字号，比如 \dfrac 命令把分数的字号设置为独立公式中的大小，而 \tfrac 命令则把字号设为行间公式中的大小。

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

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

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

例 4.3: 分数

4.2.4 运算符

有些小运算符例如 + - * / = 等可以直接输入，另一些则需要特殊命令 (见 例 4.4) 。更多的数学符号可参考 Scott Pakin 的符号列表[3]。

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

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

例 4.4: 小运算符

和、积、极限、积分等大运算符用 \sum \prod \lim \int 等命令表示 (见 例 4.5) ，它们的上下标在行间公式中被压缩，以适应行高。我们也可以用 \limits 和 \nolimits 命令显式指定是否压缩上下标。

```
$ \sum_{i=1}^n i\quad \prod_{i=1}^n\quad
    \lim_{x\to0}x^2\quad \int_a^b x^2 dx $\\
$ \sum\limits_{i=1}^n i\quad \prod\limits_{i=1}^n\quad
    \lim\limits_{x\to0}x^2\quad \int\limits_a^b x^2 dx $
\[ \sum_{i=1}^n i\quad \prod_{i=1}^n\quad
    \lim_{x\to0}x^2\quad \int_a^b x^2 dx \]
\[ \sum\nolimits_{i=1}^n i\quad
    \prod\nolimits_{i=1}^n\quad
    \lim\nolimits_{x\to0}x^2\quad
    \int\nolimits_a^b x^2 dx \]
```

$\sum_{i=1}^n i \quad \prod_{i=1}^n \quad \lim_{x\to0} x^2 \quad \int_a^b x^2 dx$
$\sum\limits_{i=1}^n i \quad \prod\limits_{i=1}^n \quad \lim\limits_{x\to0} x^2 \quad \int\limits_a^b x^2 dx$

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

$$\sum\nolimits_{i=1}^n i \quad \prod\nolimits_{i=1}^n \quad \lim\nolimits_{x\to0} x^2 \quad \int\nolimits_a^b x^2 dx$$

例 4.5: 大运算符

部分追求完美的同学可能会觉得积分公式末尾的积分变量 dx 改成 $\mathrm{d}x$ 比较好看；另外积分函数和积分变量之间需要拉开点距离。那么我们可以用 例 4.6 中的方法自己定义一个积分变量命令。

```
\newcommand{\myd}{\;\mathrm{d}}
\[ \int x dx\quad \int x \myd x \]
```

$$\int x dx \quad \int x \,\mathrm{d}x$$

例 4.6: 积分变量

多重积分如果用多个 `\int` 来输入的话，积分号之间的距离会过宽。正确的方法是用 `\iint \iiint \iiiint \idotsint` 等命令输入。从 例 4.7 中我们可以看到两种方法的差异。

```
\[ \int\int\quad \int\int\int\quad
    \int\int\int\int\quad \int\dots\int \]
\[ \iint\quad \iiint\quad \iiiint\quad \idotsint \]
```

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

例 4.7: 多重积分

4.2.5 箭头

表 4.3 给出了一些箭头的输入方法。`\xleftarrow` 和 `\xrightarrow` 命令生成的箭头可以根据内容自动调整长度（见 例 4.8）。

表 4.3: 箭头

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

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

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

例 4.8: 可扩展箭头

4.2.6 注音和标注

表 4.4 列出一些数学注音符号 (accent)，表 4.5 列出一些长的标注符号。

表 4.4: 数学注音符号

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

表 4.5: 长标注符号

$\overline{xxx}$	\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}	$\widehat{xxx}$	\widehat{xxx}
$\underrightarrow{xxx}$	\underrightarrow{xxx}	$\widetilde{xxx}$	\widetilde{xxx}

4.2.7 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 LaTeX 中 |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 等命令来调整其大小。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。

```
\[ \Bigg(\bigg(\Big(\big((x)\big)\Big)\bigg)\Bigg)\quad
\Bigg[\bigg[\Big[\big[[x]\big]\Big]\bigg]\Bigg]\quad
\Bigg\{\bigg\{\Big\{\big\{\{x\}\big\}\Big\}\bigg\}\Bigg\}
\]\[
\Bigg\langle\bigg\langle\Big\langle\big\langle\langle x
\rangle\big\rangle\Big\rangle\bigg\rangle\Bigg\rangle\quad
\Bigg\lvert\bigg\lvert\Big\lvert\big\lvert\lvert x
\rvert\big\rvert\Big\rvert\bigg\rvert\Bigg\rvert\quad
\Bigg\lVert\bigg\lVert\Big\lVert\big\lVert\lVert x
\rVert\big\rVert\Big\rVert\bigg\rVert\Bigg\rVert \]
```

例 4.9: 分隔符

4.2.8 省略号

省略号用 \dots \cdots \vdots \ddots 等命令表示。\dots 和 \cdots 的纵向位置不同；前者一般用于有下标的序列。

```
\[ x_1,x_2,\dots,x_n\quad 1,2,\cdots,n\quad
    \vdots\quad \ddots \]
```

$$x_1, x_2, \dots, x_n \quad 1, 2, \cdots, n \quad \vdots \quad \ddots$$

例 4.10: 省略号

4.2.9 空白间距

在数学模式中，我们可以用 表 4.6 中的命令生成合适的空白间距，注意负间距命令 \! 可以用来减小间距。

表 4.6: 空白间距

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

4.3 矩阵

数学模式下可以用 array 环境（见 例 4.11）来生成矩阵，它提供了外部对齐和列对齐的控制参数。外部对齐是指整个矩阵和周围对象的纵向关系，有三种方式：居顶、居中 (缺省) 、居底，分别用 t、c、b 来表示；列对齐也有三种方式：居左、居中、居右，分别用 l、c、r 表示。\\ 和 & 用来分隔行和列。

其语法如下：

```
\begin{array}[外部对齐]{列对齐}
    行列内容
\end{array}
```

amsmath 的 pmatrix、bmatrix、Bmatrix、vmatrix、Vmatrix 等环境可以在矩阵两边加上各种分隔符，但是它们没有对齐方式参数（见 例 4.12）。

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

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

例 4.11: 矩阵

```
\[ \begin{pmatrix} a&b\\c&d \end{pmatrix} \quad
    \begin{bmatrix} a&b\\c&d \end{bmatrix} \quad
    \begin{Bmatrix} a&b\\c&d \end{Bmatrix} \quad
    \begin{vmatrix} a&b\\c&d \end{vmatrix} \quad
    \begin{Vmatrix} a&b\\c&d \end{Vmatrix} \]
```

$$\begin{pmatrix} a & b \\ c & d \end{pmatrix} \quad \begin{bmatrix} a & b \\ c & d \end{bmatrix} \quad \begin{Bmatrix} a & b \\ c & d \end{Bmatrix} \quad \begin{vmatrix} a & b \\ c & d \end{vmatrix} \quad \begin{Vmatrix} a & b \\ c & d \end{Vmatrix}$$

例 4.12: 更多矩阵

`\smallmatrix` 命令可以生成行间矩阵（见 例 4.13）。

```
Marry has a little matrix $ ( \begin{smallmatrix}
a&b\\c&d \end{smallmatrix} ) $.
```

Marry has a little matrix $\left(\begin{smallmatrix} a & b \\ c & d \end{smallmatrix} \right)$.

例 4.13: 行间矩阵

4.4 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` 提供的一些多行公式环境。

4.4.1 长公式

无须对齐的长公式可以使用 `multline` 环境（见 例 4.14）。需要对齐的长公式可以使用 `split` 环境（见 例 4.15），它本身不能独立使用，必须包含在其它数学环境内，因此也称作次环境。它用 `\\` 和 `&` 来分行和设置对齐的位置。

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

$$x = a + b + c + d + e + f + g \tag{4.2}$$

例 4.14: 无对齐长公式

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

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

例 4.15: 对齐长公式

4.4.2 公式组

不需要对齐的公式组可以使用 `gather` 环境（见 例 4.16），需要对齐的公式组用 `align` 环境（见 例 4.17）。

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

$$a = b + c + d \tag{4.3}$$
$$x = y + z \tag{4.4}$$

例 4.16: 无对齐公式组

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

$$a = b + c + d \tag{4.5}$$
$$x = y + z \tag{4.6}$$

例 4.17: 对齐公式组

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

4.4.3 分支公式

分段函数通常用 `cases` 次环境写成分支公式（见 例 4.18）。

4.5 定理和证明

`\newtheorem` 命令可以用来定义定理之类的环境，其语法如下。

```
\[ y=\begin{cases}
    -x,\quad x\leq 0 \\
    x,\quad x>0
\end{cases} \]
```

$$y = \begin{cases} -x, & x \leq 0 \\ x, & x > 0 \end{cases}$$

例 4.18: 分支公式

语法：{环境名}[编号延续]{显示名}[编号层次]

下面的代码定制了四个环境：定义、定理、引理和推论，它们都在一个 section 内统一编号，而引理和推论会延续定理的编号。我们在 例 4.19 中定制了一些环境后，可以像 例 4.20 那样使用它们。

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

例 4.19: 定制定理类环境

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

定义 4.5.1. *Java* 是一种跨平台的编程语言。

```
\begin{theorem}
咖啡因可以刺激人的中枢神经。
\end{theorem}
```

定理 4.5.1. 咖啡因可以刺激人的中枢神经。

```
\begin{lemma}
茶和咖啡都会使人的大脑兴奋。
\end{lemma}
```

引理 4.5.2. 茶和咖啡都会使人的大脑兴奋。

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

推论 4.5.3. 晚上喝咖啡可能会导致失眠。

例 4.20: 使用定理类环境

amsthm 宏包提供的 proof 环境（见 例 4.21）可以用来输入证明，它会在证明结尾加一个 QED 符号 [2]。

[2]拉丁语 quod erat demonstrandum 的缩写。

```
\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。 □

例 4.21: 证明

4.6 数学字体

和文本模式类似，我们在数学模式下也可以选用不同的字体样式 (见 表 4.7) 。`\mathbb` 和 `\mathfrak` 需要 `amsfonts` 宏包，`\mathscr` 需要 `mathrsfs` 宏包。

表 4.7: 数学字体

缺省	$ABCXYZ$	`\mathbf`	$\mathbf{ABCXYZ}$
`\mathrm`	ABCXYZ	`\mathit`	ABCXYZ
`\mathsf`	ABCXYZ	`\mathbb`	$\mathbb{ABCXYZ}$
`\mathtt`	$\mathtt{ABCXYZ}$	`\mathfrak`	$\mathfrak{ABCXYZ}$
`\mathcal`	$\mathcal{ABCXYZ}$	`\mathscr`	$\mathscr{ABCXYZ}$

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/dp/0387322892/.

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

广告位招租

第五章　插图

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

当年 Knuth 开发 TEX 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 TEX 里留了一个后门：`\special` 命令，让后面的驱动自行决定怎样处理图形。

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，要我改改当作教学辅助软件出版，但是当时手头没有 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 5.1 节讨论 LATEX 所用图形格式以及图形的优化、转换和处理，5.2 节介绍怎样插入图形，5.3 节简介矢量绘图。接下来的三章 6–8 会分别讨论怎样使用 METAPOST, PSTricks 和 PGF。

5.1　图形概览

5.1.1　图形格式

LATEX 支持点阵图形格式 JPEG 和 PNG，也支持矢量格式 EPS 和 PDF [1]。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。

1980 年代中后期，PostScript 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。然而它实在太强大，人们担心嵌入文档的 PostScript 会搞破坏，于是就产生了戴着手铐的 Encapsulated PostScript (EPS)。

[1] EPS 和 PDF 中也可以嵌入点阵图形，但是它们本身还是矢量格式。

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。早年间人们得到 DVI 后通常会把它转换为 PostScript，所以 EPS 就成了 LaTeX 的标准图形格式。

5.1.2 Driver 的口味

dvips

`dvips` 喜欢 PostScript，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。但是它很固执，坚持按缺省分辨率 72 PPI 计算图形尺寸，并认为所有图形都是灰度图。为了避免麻烦，包老师还是劝你把这些图形格式转换为 EPS。

pdflatex

`pdflatex` [2] 支持 JPEG、PNG 和 PDF，不支持 EPS。传说它不支持 EPS 的原因是 PostScript 解释器的版权问题。包老师认为这种说法不可信，因为 1997 年 pdfTeX 面世时 PostScript 已经被 PDF 赶超，Adobe 与其保护 PostScript 还不如保护 PDF。

LaTeX 有两个宏包 `epstopdf` 和 `pst-pdf` 可以实时地 (on the fly) 把 EPS 转换为 PDF [3]。然而前者有安全漏洞，后者用法繁琐，用户最好还是用其他软件事先把 EPS 转为 PDF。

dvipdfm(x)

`dvipdfm` 支持 JPEG、PNG 和 PDF，不支持 EPS，但是它可以实时地调用 Ghostscript 把 EPS 转为 PDF。`dvipdfmx` 对上述图形格式的支持有所增强，还增加了对 BMP 的支持。

xdvipdfmx

XeLaTeX 的缺省驱动 `xdvipdfmx` 直接支持 BMP、JPEG、PNG、EPS 和 PDF。所以从图形格式支持的角度来讲，`xdvipdfmx` 比 `dvips`、`pdflatex` 和 `dvipdfmx` 都好。传说 `xdv2pdf` 驱动还支持 GIF、PICT、PSD、SGA、TGA、TIFF 等格式，可惜只能在 Mac OS X 上用。

[2] `pdflatex` 包含编译和驱动两种功能，所以这里也就把它划到驱动一边。

[3] 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。

5.1.3 图形优化

矢量图形的一个优点是可以无限缩放，而输出质量不变。图形尺寸对矢量图形而言意义不大。描述矢量图形所需数据较少，所以其文件体积一般也较小。

而点阵图形是以像素 (pixel) 为单位描述、存储的，图形尺寸越大，文件体积就越大。当然影响文件体积的还有色彩深度、压缩算法等因素。

人们一般希望用较小的文件体积获取较好的输出效果，这样就需要优化图形尺寸和色彩。

图形尺寸

点阵图形的像素是一种相对尺寸，其实际尺寸等于像素除以分辨率 (resolution)，最常用的分辨率单位是像素 / 英寸 (pixels per inch，PPI)。在古时候 PPI 也常和点 / 英寸 (dots per inch, DPI) 混用；现在人们倾向于认为 PPI 是图形的分辨率单位，而 DPI 是硬件设备 (比如显示器或打印机) 输出的分辨率单位。通常横向和纵向分辨率相同，可以写成一个数字。

比如有一幅 100 x 150 像素的点阵图形 (图 5.1)，其分辨率为 100 PPI。在输出时，它的缺省尺寸就是 1in x 1.5in。如果我们将它强制输出为 2in x 3in (图 5.2)，那么其实际分辨率就降为 50 PPI。

图 5.1: 缺省输出尺寸

图 5.2: 强制放大输出尺寸

假设输出分辨率是 100 DPI，像素和输出点就一一对应；如果输出分辨

率是 300 DPI，每个像素实际输出为 3 x 3 个点；如果输出分辨率是 50 DPI，每 2 x 2 个像素才享受到一个输出点的待遇。

当图形分辨率和输出分辨率不一致时，就会有一个重新采样 (resampling) 的过程；从高分辨率到低分辨率叫下采样 (downsampling) ，反之叫上采样 (upsampling) 。重新采样的插值 (interpolation) 算法有很多，其中常用的有最近像素 (nearest neighbor) 、双线性 (bilinear) 、双三次 (bicubic) 、兰索斯 (Lanczos) 等。前两种速度快，但是效果差；后两种效果好，但是速度慢。追求完美的雷人自然要选择 Lanczos。

一般而言，高分辨率图形配合高分辨率输出设备会产生高质量效果，低分辨率图形配合低分辨率设备会产生低质量效果；高分辨率图形遇到低分辨率设备会形成浪费；低分辨率图形遇到高分辨率设备呢，得看插值效果，但是最好委高估机器的智能。雷人追求的是高质量文档，所以输出设备的分辨率大家暂时就甭操心了。

当输出尺寸一定时，图形分辨率越高需要的像素就越多，图形文件体积就越大。那么点阵图形的分辨率多少比较合适呢？一般认为在屏幕上阅读需要 72 PPI，考虑到放大 150 PPI 应该够了，而高质量打印需要 300 PPI。

假设我们要在 LaTeX 文档中嵌入一幅图形，如果是通栏，宽度就是 4.8–5.4in (文档缺省宽度取决于字体大小) 。那么如果仅用于屏幕阅读，原始图形的宽度 400px 足够了；如要放大阅读或输出打印则分别需要 800px 和 1600px。

点阵图形尺寸相关的基本编辑操作有以下几种：裁剪 (crop) 、改尺寸 (resize) 、改分辨率。

1. 裁剪时像素自然会变少，分辨率不变，缺省输出尺寸也就变小。图 5.1 其实就是从一幅 2048 x 1536 像素的大图裁剪、缩小而来，因为原图太大，图 5.3 把它强制输出为 4in x 3in。

2. 改尺寸时像素改变，分辨率不变，缺省输出尺寸相应改变，这个过程需要重新采样。比如把 图 5.1 resize 到 200 x 300 像素，分辨率还是 100 PPI，那么缺省输出尺寸就变成 2in x 3in (图 5.4) 。

3. 单纯更改图形文件分辨率时，像素不变，缺省输出尺寸相应改变。把 图 5.1 分辨率设为 200 PPI，像素还是 100 x 150，那么缺省输出尺寸就变成 0.5 x 0.75in (图 5.5) 。

4. 改尺寸和改分辨率可以结合使用。把 图 5.1 的尺寸改为 200 x 300 像素，分辨率改为 200 PPI，缺省输出尺寸就还是 1in x 1.5in (图 5.6) 。

综上所述，点阵图形的信息量取决于像素。图形文件的分辨率只是“建议”缺省输出尺寸，并不影响图形质量。上述操作中裁剪和改尺寸比较实用，改分辨率没有实质意义。改尺寸一般也只能从大改小。如果从小改大的话，插补出来的像素比起原装的还是要差一些。

图 5.3: 原始图形

图 5.4: 改尺寸

图 5.5: 改分辨率

图 5.6: 改尺寸和分辨率

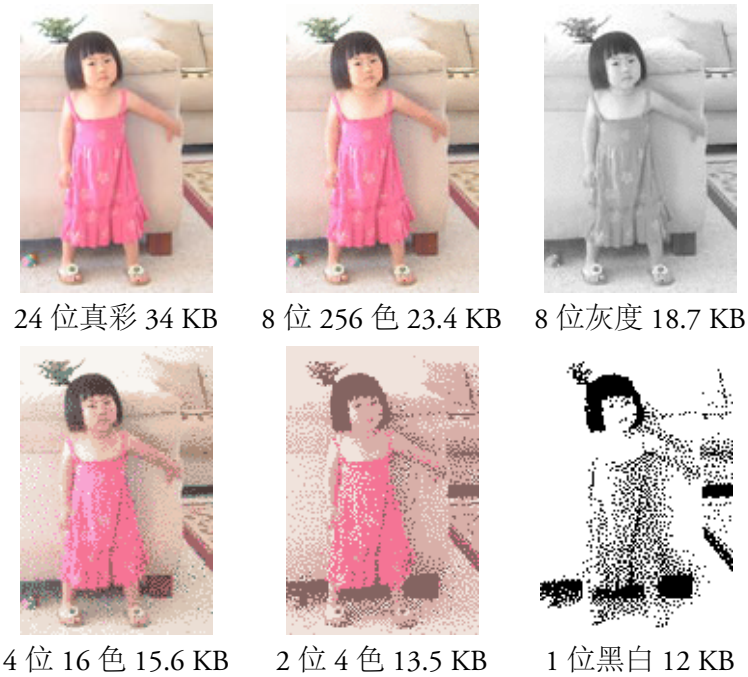
需要指出的是上述操作在不同的软件里名称不同，比如改尺寸，相当于

ImageMagick 中的 resize 和 Adobe PhotoShop 中的 resample；改分辨率，相当于 ImageMagick 中的 density 和 PhotoShop 中的 resize；上面第四种混合操作，在 ImageMagick 叫 resample。包子曰：道可道，非常道。名可名，非常名。

色彩深度

色彩深度 (color depth) 是每一个像素所用颜色的位数。比如一位可以表示两种颜色，通常是黑白；两位可以表示四色，最早用于 CGA 显卡；四位 16 色，用于 EGA 显卡；八位 256 色，用于 VGA 显卡；16 位 65,536 色，又称高彩；24 位 16,777,216 色，又称真彩；30–48 位称为深彩。

色深位数越高越逼真，文件体积也就越大。一般照片可以用 24 位，人工图像用八位足矣，图标之类的小图形可以考虑更少位数。从 图 5.7 我们可以看到各种色深的效果和文件体积，它们都是 PNG 格式。



24 位真彩 34 KB　8 位 256 色 23.4 KB　8 位灰度 18.7 KB

4 位 16 色 15.6 KB　2 位 4 色 13.5 KB　1 位黑白 12 KB

图 5.7: 色彩深度

我们一般也只能把图形的色深从高改低，从而减小图形文件和最终文档的体积。反过来把低色深从低改高，属于逆势而为，或遭天谴。

5.1.4 图形转换和处理

按说有了 XeLaTeX 之后雷人已经基本上不再需要转换图形格式，只是有时出版社会指定使用某几种图形格式；另外我们常常需要优化图形，否则直接嵌入文档有点浪费。这时图形处理软件就派上了用场。

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”。

第一章简介中提到的 Ghostscript 不仅包含 RIP，还提供 PostScript、EPS 和 PDF 等文件格式的转换功能。通常人们会通过 Ghostscript 的一个图形前端来调用它的功能，它最常用的前端是 GSview。

通用图像处理软件种类繁多。如果你喜欢命令行界面包老师推荐 ImageMagick，喜欢图形界面的可以试试 Paint.NET，喜欢凌乱的界面而且内存多得用不完的可以试试 GIMP。其他收费软件有悖于自由软件的精神，这里不提也罢。

ImageMagick

ImageMagick 包含多个命令行程序，其中最常用的是 `convert`。下面的命令把 BMP 转换为 PNG，据说 ImageMagick 可以识别 100 多种格式。

```
convert fig.bmp fig.png
```

Windows 有个用来转换分区格式的同名程序。所以我们在 Windows 下使用 ImageMagick 时，需要写全路径，或者在 `PATH` 环境变量里把 ImageMagick 的路径放到 `system32` 前面。

图形尺寸相关操作命令如 例 5.1 所示。第一行命令裁剪并且转换格式，截取从 (10,10) 开始 300 x 200 像素的图像，原点在左上；第二行裁剪并缩放；第三行缩放到 300 x 200 像素范围内，保持长宽比；第四行强制缩放到给定尺寸，不考虑长宽比；第五行把分辨率改为 300 PPI；第六行把分辨率改为 300 PPI，像素增加，缺省输出尺寸维持不变。

```
convert fig.bmp -crop 300x200+10+10 fig.jpg
convert fig.jpg -crop 300x200+10+10 -resize 30x20 fig1.jpg
convert fig.jpg -resize 300x200 fig1.jpg
convert fig.jpg -resize !300x200 fig1.jpg
convert fig.jpg -density 300 fig1.jpg
convert fig.jpg -resample 300 fig1.jpg
```

例 5.1: ImageMagick 尺寸操作

图形色深相关命令如 例 5.2 所示。第一行命令把图形转为 8 位 256 色，第二行转为 8 位灰度，第三行转为 4 位 16 色，第四行转为 2 位 4 色，第五行转为 1 位黑白。

```
convert fig.jpg -colors 256 png8:fig8.png
convert fig.jpg -colorspace gray png8:fig8g.png
convert fig.jpg -colors 16 png8:fig4.png
convert fig.jpg -colors 4 png8:fig2.png
convert fig.jpg -monochrome fig1.png
```

例 5.2: ImageMagick 色深操作

ImageMagick 功能强大，参数选项很多，这里只能蜻蜓点水。它有一个缺点，缩小图像做缩略图时不是很清晰；也许可以调整参数改善清晰度。我用过的软件中 ACDSee 做的缩略图最清晰，但它是收费软件。

其他格式转为 EPS

有很多软件都可以把点阵图像转换为 EPS，比如 ImageMagick 和 GIMP，以及 a2ping/sam2p、bmeps、jpeg2ps、sam2p 等。

PostScript 从 Level 2 开始才支持点阵图像压缩，所以在把其他格式转为 EPS 时应尽量使用 Level 2 或 3，否则输出的 EPS 会很大。

ImageMagick 转换 EPS 的方法如下。如果是 BMP 文件，最好先压缩成 JPEG 或 PNG，再转为 EPS，这样生成的 EPS 会比较小。我猜 EPS 的缺省压缩算法可能不如 JPEG 和 PNG。

```
convert fig.png eps3:fig.eps
```

另一种方法是用虚拟打印机生成 EPS，它的优点是可以把几乎所有文件“打印”成 EPS。包老师推荐 Bullzip PDF Printer，它可以把各种文件打印成 PS、EPS、PDF、BMP、JPEG、PCX、PNG、TIFF 等格式。

用合适的软件打开原始文件，打印到 Bullzip PDF Printer。在 General 标签页把 Format 设置为 EPS，点 Save 按钮就会得到 EPS。

用其他 PostScript 打印机的驱动程序也可以生成 EPS，只是稍繁琐。因为它首先要把原始文件打印生成 PS，再用 GSview 打开转为 EPS。此方法已被包老师淘汰，考古者可参考 lnotes 第一版[1]4.1.3 节。

其他格式转为 PDF

我们可以先把其他图像格式转为 EPS，再用 Ghostscript 提供的 `ps2pdf` 程序把它转为 PDF。

```
ps2pdf -dEPSCrop fig.eps fig.pdf
```

我们也可以用 PDF 虚拟打印机直接把其他图像文件打印为 PDF，只是这样生成的 PDF 没有裁剪空白边。

另外 ImageMagick 和 LaTeX 附带的 `epstopdf` 程序 [4] 也都可以把 EPS 转为 PDF，只是前者效果不好，后者不稳定。

5.2 插入图形

5.2.1 范围框

由于历史原因，`latex` 编译程序不能提取 JPEG、PNG 等点阵图形的尺寸信息，所以它在处理这些图形文件时需要范围框 (bounding box) 。`pdflatex` 和 `xelatex` 的用户可以跳过本小节，因为它们出现的比较晚，有机会了解这些图形格式。

上文提到尺寸对矢量图形而言意义不大，然而 EPS 是一种嵌入图形格式，有个缺省尺寸还是比较方便，这就是它的范围框。因为 EPS 最先被 LaTeX 支持，范围框的概念就被沿用了下来。

EPS 的范围框如下，其中前两个参数是图形左上角的坐标 (通常就是原点) ，后两个参数是右下角的坐标，缺省长度单位是 bp。为什么这幅 EPS 左上角不从 (0,0) 开始呢，也许是为了裁剪空白，或者想隐藏点什么东西。

```
%!PS-Adobe-3.0 EPSF-3.0
%%BoundingBox: 5 5 105 105
```

有了范围框，`latex` 在编译源文件时就可以为插图预留空间；它输出的 DVI 只记录图形尺寸和文件名，因为具体的图形处理由后面的驱动负责。找不到范围框时，`latex` 就会报错，

```
! LaTeX Error: Cannot determine size of graphic in fig.png
    (no BoundingBox).
```

有两种方法可以为点阵图形提供范围框：一种是准备一个单独的范围

[4] 这个命令行程序和上面提到的 `epstopdf` 宏包是两样东西。

框文件，另一种是在插入图形时加范围框参数。如果必须使用 latex，包老师推荐用第二种方法，因为文件多了不便管理。

dvipdfm 附带的 ebb 程序可以检查 JPEG 和 PNG，生成范围框文件。比如下面的命令会生成一个 fig.bb 文件。

```
ebb fig.png
```

然而 ebb 有个缺点，它懒得理会图像文件的真正分辨率，直接用 100 PPI 来计算，这样的做法很粗鲁。追求完美的雷人可以自行计算尺寸，自己写范围框文件或插入图形时加上范围框参数。bp 值 = 像素/分辨率 * 72。

5.2.2 基本命令

上文提到 Knuth 留下了后门 \special，但是直接用它来插入图形不够含蓄优雅，于是 LaTeXv2.09 推出了 epsf 和 psfig 宏包。之后 David P. Carlisle (1961–) [5] 和 Rahtz 推出了面向 LaTeX 2_ε 的 graphics 和 graphicx 宏包；后者基于前者，语法更简单，功能更强大，所以一般推荐用它。

插图命令基本用法如下，

```
\usepackage[dvipdfm]{graphicx}
\includegraphics[bb=0 0 300 200]{fig.png}
```

引用 graphicx 宏包时可加驱动选项，使用 latex 时缺省驱动是 dvips，dvipdfm(x) 用 dvipdfm；pdflatex 和 xelatex 分别用 pdftex 和 xetex，但是它们知道驱动就是自己，其实不用加该选项。

使用 latex 时，如果事先没有生成 .bb 文件的话，需要或加范围框参数。pdflatex 和 xelatex 不要该参数，加上反而可能误事。

5.2.3 图形操作

\includegraphics 命令有一些参数选项 (见 表 5.1) 可以用于缩放、旋转、裁剪等图形操作，简要说明如下：

- 如果不设置任何尺寸参数，latex 按范围框处理；dvipdfm(x) 和 pdflatex 按缺省输出尺寸处理。如果图形文件缺少 PPI，后面这两位会按 72 PPI 计算输出尺寸。

[5] 1995 年曼彻斯特大学数学博士，剑桥博士后，1998 年加入数字算法公司 (Numerical Algorithms Group)。

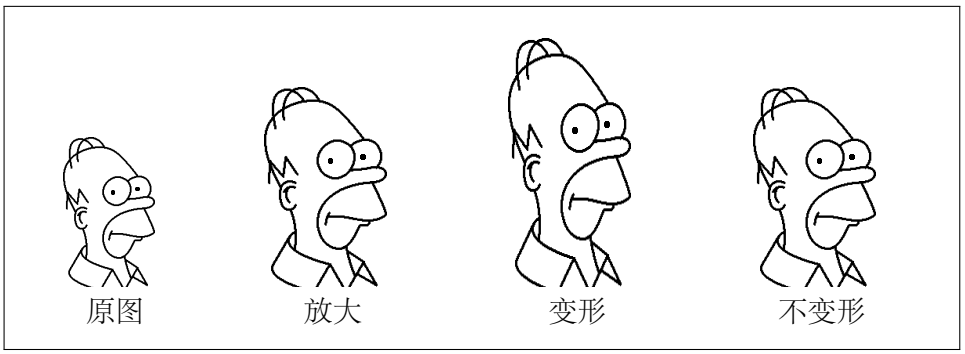
- 我们可以用 `scale` 来缩放输出尺寸，但是如果范围框或 PPI 不准，结果会出乎意料。所以包老师建议还是用绝对尺寸参数比较好。

- XeLaTeX 目前不支持裁剪，用户只好自己先行处理。

表 5.1: 图形操作选项

`width=x,height=y`	宽度和高度，绝对尺寸，可用任意长度单位。
`scale=s`	缩放比。绝对尺寸和缩放比用一种即可，同时使用两者，绝对尺寸起作用。
`keepaspectratio`	保持图形比例。宽度和高度通常设置一个即可，否则图形比例会失调，除非再加上此选项，这样图形宽度和高度都不超过指定参数。
`angle=a`	逆时针旋转角度，单位是度。
`origin=hv`	旋转中心，缺省在左下。水平和垂直方向分别可选左、中、右和上、中、下，用 l、c、r 和 t、c、b 表示。
`totalheight=h`	总高度，最高、最低两点之间垂直距离。
`viewport=x1 y1 x2 y2`	可视区域左上角和右下角坐标，缺省单位 bp。
`trim=l b r t`	左、下、右、上四边裁剪值，缺省单位 bp。
`clip`	是否真正裁剪，配合 viewport 或 trim 使用。如不使用此参数，被裁剪部分依然显示，会和插图周围内容重叠。
`page=n`	选页，用于多页图形文件。

```
\includegraphics[width=60pt]{homer.pdf}
\includegraphics[width=80pt]{homer.pdf}
\includegraphics[width=80pt,height=100pt]{homer.pdf}
\includegraphics[width=80pt,height=100pt,keepaspectratio]{
    homer.pdf}
```

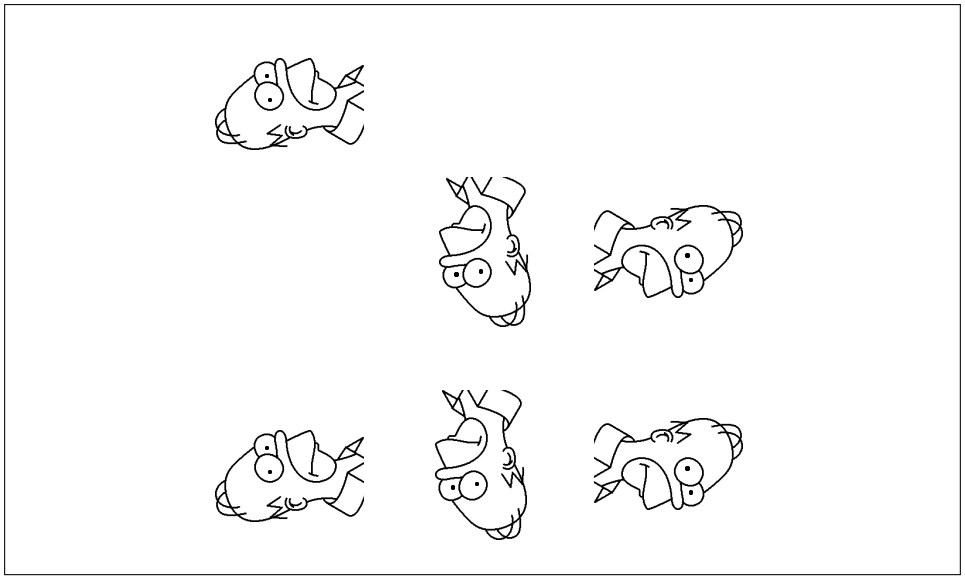


例 5.3: 图形缩放

插图的缩放操作见 例 5.3，其中第三幅因为同时使用了宽度和高度选项而产生了变形。

插图的旋转操作见 例 5.4，其中前三幅图的旋转中心在左下角，后三幅的在图中心。

```
\includegraphics[angle=90]{homer.pdf}
\includegraphics[angle=180]{homer.pdf}
\includegraphics[angle=270]{homer.pdf} \\
\includegraphics[angle=90,origin=c]{homer.pdf}
\includegraphics[angle=180,origin=c]{homer.pdf}
\includegraphics[angle=270,origin=c]{homer.pdf} \\
```



例 5.4: 图形旋转

5.2.4 文件名和路径

若想省略文件后缀或路径名，可以使用 例 5.5 中的命令。其中第一行指定后缀列表让编译程序自行查找；第二行指出未知后缀的都是 EPS；后三行设置缺省搜索路径，分别使用了绝对路径、相对路径、多个路径。注意文件名和路径名都不能有空格；路径名分隔符最好用正斜杠 /，这样可以在多种操作系统上通用；路径名要用 / 结尾。

对于文件后缀，包老师认为少敲几个字符省不了多少气力，只会让电脑多花时间搜索，与低碳环保之精神相抵触。对于路径，如果数量较少，而且不同路径下没有重复文件名的话，可以设置搜索路径。

```
\DeclareGraphicsExtensions{.eps,.mps,.pdf,.jpg,.png}
\DeclareGraphicsRule{*}{eps}{*}{}
\graphicspath{{c:/secret_garden/}}
\graphicspath{{./img/}}
\graphicspath{{one_little/}{two_little/}{three_little_
    indians/}}
```

例 5.5: 插图文件名和路径

5.2.5 figure 环境

插图通常需要占据大块空间，所以在文字处理软件中用户经常需要调整插图的位置。`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 (float)，下一章里还会介绍表格浮动环境。

在 例 5.6 中，`htbp` 选项用来指定插图的理想位置，这几个字母分别代表 here, top, bottom, float page，也就是就这里、页顶、页尾、浮动页 (专门放浮动环境的单独页面)。

```
\begin{figure}[htbp]
\centering
\includegraphics{myphoto.jpg}
\caption{有图有真相}
\label{fig:myphoto}
\end{figure}
```

例 5.6: `figure` 环境

我们可以使用这几个字母的任意组合，四个母都写上表示放哪里都无所谓；一般不推荐单独使用 `h`，因为 LaTeX 自以为它的排版算法是最完美的，不愿意被束缚手脚。

`\centering` 用来使插图居中；`\caption` 命令设置插图标题，LaTeX 会自动给浮动环境的标题加上编号。注意 `\label` 应该放在标题之后，否则引用时指向的是前一个结构对象。

5.2.6 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 (见 例 5.7)。

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

图 5.8: 反清复明

例 5.7: 并排摆放，共享标题

并排摆放，各有标题

如果想要两幅并排的插图各有自己的标题，可以在 figure 环境中使用两个 minipage 环境，每个里面插入一幅图 (见 例 5.8) 。不用 minipage 的话，因为插图标题的缺省宽度是整个行宽；两幅插图就会上下排列。

```
\begin{figure}[htbp]
\centering
\begin{minipage}{60pt}
\centering
    \includegraphics{left.pdf}
    \caption{清明}
\end{minipage}
\hspace{10pt}%
\begin{minipage}{60pt}
\centering
    \includegraphics{right.pdf}
    \caption{反复}
\end{minipage}
\end{figure}
```

图 5.9: 清明　图 5.10: 反复

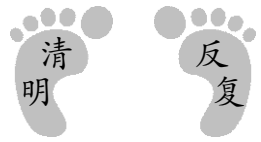
例 5.8: 并排摆放，各有标题

\caption 命令会把环绕它的 minipage 环境“变成” figure 环境。代码第 2、4、11 行有三个 \centering 命令，第一个使得两个 minipage 整体居中，后两个分别使得 minipage 中的内容居中。第 14 行的 \hspace 命令用来调整两个 minipage 之间的距离。

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，并且各有自己的子标题，可以使用 Steven D. Cochran [6] 开发的 `subfig` 宏包。它提供的 `\subfloat` 命令用法见 例 5.9，总图和子图可以分别有标题和引用。

```
\begin{figure}[htbp]
\centering
\subfloat[左脚清明]{
    \label{fig:subfig_a}
    \includegraphics{left.pdf}
}
\hspace{10pt}%
\subfloat[右脚反复]{
    \label{fig:subfig_b}
    \includegraphics{right.pdf}
}
\caption{反清复明}
\label{fig:subfig}
\end{figure}
```



(a) 左脚清明　(b) 右脚反复

图 5.11: 反清复明

例 5.9: 并排摆放，共享标题，各有子标题

改进的子图方法

`\subfloat` 命令缺少宽度参数，而子标题最多只能和子图一样宽，太长的话会出现折行。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（见 例 5.10）。

想了解更多子图功能请参考 `subfig` 宏包手册[2]，更多 LaTeX 插图功能，请参考 Keith Reckdahl 的 epslatex[3]。

[6] 卡耐基梅隆计算机系前高级系统研究员，现在可能退休了。

```
\begin{figure}[htbp]
\centering
\subfloat[左脚清明]{
    \label{fig:improved_subfig_a}
    \begin{minipage}[t]{60pt}
        \centering
        \includegraphics{left.pdf}
    \end{minipage}
}
\subfloat[右脚反复]{
    \label{fig:improved_subfig_b}
    \begin{minipage}[t]{60pt}
        \centering
        \includegraphics{right.pdf}
    \end{minipage}
}
\caption{反清复明}
\label{fig:improved_subfig}
\end{figure}
```

(a) 左脚清明　(b) 右脚反复

图 5.12: 反清复明

例 5.10: 改进的子图方法

5.3 矢量绘图

5.3.1 色彩模型

在图形、表格甚至文字中我们都可以使用色彩，所以应该对色彩模型有所了解。一般而言，色彩模型分两大类：一种是广泛应用于照相、摄影、电视和电脑的 RGB 三原色模型（图 5.13），另一种是用于彩色印刷的 CMYK 四分色模型（图 5.14）。前者是加色模型，后者是减色模型。

RGB 模型认为色彩世界原本是无任何光线的黑色，如果把红、绿、蓝三种颜色的光以不同比例叠加，就可以得到任何颜色。比如红、绿叠加得到黄，绿、蓝叠加得到青，蓝、红叠加得到洋红，红、绿、蓝叠加得到白。

在 24 位真彩模型中，三原色各使用一个 8 位，取值从 0 到 255。

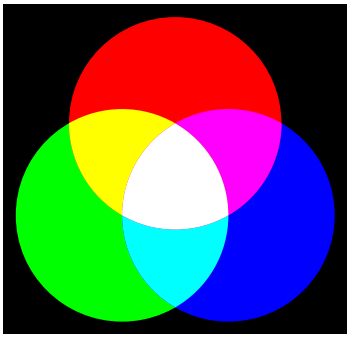


图 5.13: RGB 模型

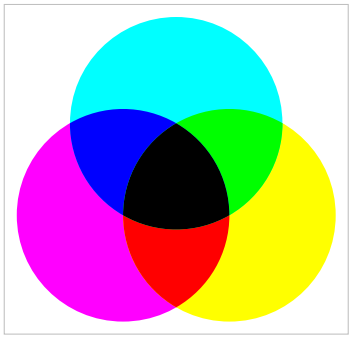


图 5.14: CMYK 模型

比如红、绿、蓝、黑、白分别为：(255,0,0)、(0,255,0)、(0,0,255)、(0,0,0)、(255,255,255)。如果使用 HTML 中常用的 16 进制表示，就是 FF0000、00FF00、0000FF、000000、FFFFFF。

RGB 模型用的是笛卡尔坐标，色彩的变化对人眼来说不够连续，人们就提出把它的坐标系改为圆柱坐标，这就是 HSL 和 HSV 模型。基于这种模型，人们可以用色轮选色（图 5.15），并美其名曰色轮常转。

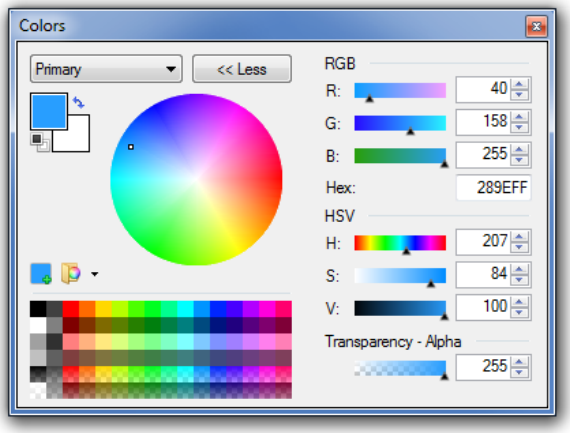


图 5.15: 选色

CMYK 模型认为纸张原本是白色或浅色背景的，在上面印刷某种颜料就会减少这种颜色光的反射，看上去就会是它的补色。印青色就会得到红色，印洋红得到绿，印黄得到蓝，青、洋红、黄三种颜色都印上就会得到黑色。彩色印刷有分色和套色过程，如果图形上有黑色直接拿黑颜料印刷会减少成本。CMYK 中的字母 K 就代表 key black。

在 LaTeX 中，上述模型可以用不同的模式表示。比如 RGB 模型有三种模式：10 进制的 RGB 模式，16 进制的 HTML 模式，$[0,1]$ 实数的 rgb 模式。

因为各种驱动支持不同的模式，用起来会很麻烦，color 宏包提供了驱动独立的使用界面。Uwe Kern [7] 的 xcolor 宏包更进一步，整合了 12 种色彩模式 (rgb, cmy, cmyk, hsb, Hsb, tHsb, gray, RGB, HTML, HSB, Gray, wave) ，提供了丰富的预定义颜色和命令。

预定义和自定义颜色

xcolor 宏包中预定义的颜色有：19 种基本颜色，68 种 dvips 颜色，151 种 SVG 颜色，317 种 Unix/X11 颜色。如要使用后三类颜色，引用宏包时需加相应预定义颜色集合选项：

```
\usepackage[dvipsnames]{xcolor}
\usepackage[svgnames]{xcolor}
\usepackage[x11names]{xcolor}
```

如果这几百种预定义颜色还不能满足需要，可以使用 \definecolor 命令自定义更多颜色。

语法：\definecolor{名称}{模式}{参数}

```
\definecolor{myred}{RGB}{255,0,0}
\definecolor{mygreen}{HTML}{00FF00}
\definecolor{myblue}{rgb}{0,0,1}
```

例 5.11: 自定义颜色

彩色文字

设置文字的颜色可以使用 \textcolor 命令，例 5.12 中代码第 2–4 行和第 5–7 行输出效果相同。后三行的方法又称为抛弃型颜色定义法，因为只能用一次；事先定义了名字的话还可以重用。

语法：\textcolor{名称}|[模式]{代码}{文字}

在当初电脑内存很宝贵的岁月里，引用 xcolor 宏包时不加预定义颜色集合选项就可以节省几十乃至几百个变量；当然这项节约对如今电脑的硬件配置不值一提，如果你坚持要做内存葛朗台，后三行的写法还是可以满足你变态的心理。

[7] 1993 年维尔茨堡大学 (University of Würzburg) 数学博士。

```
\textcolor{Red}{红}
\textcolor{Green}{绿}
\textcolor{Blue}{蓝}
\textcolor[RGB]{255,0,0}{红}
\textcolor[HTML]{00FF00}{绿}
\textcolor[rgb]{0,0,1}{蓝}
```

红 绿 蓝 红 绿 蓝

例 5.12: 彩色文字

彩色盒子

`\colorbox` 命令可以生成有彩色背景的盒子，它的语法和 `\textcolor` 类似。`\fcolorbox` 命令又给彩色盒子加了边框，它的第一个参数是边框的颜色。例 5.13 中使用了包老师喜欢的几种颜色，它们都来自 `svgnames`。

```
\colorbox{Lavender}{}
\colorbox{SkyBlue}{}
\colorbox{Wheat}{}
\fcolorbox{Silver}{Lavender}{}
\fcolorbox{RoyalBlue}{SkyBlue}{}
\fcolorbox{SandyBrown}{Wheat}{}
```

例 5.13: 彩色盒子

更多色彩功能可参考 `xcolor` 宏包的手册[4]。包老师敬告读者，在文档中要慎用彩色。包子曰：五音乱耳，五色炫目。

5.3.2 绘图工具概览

与 LaTeX 配套使用的矢量绘图工具中包老师较熟悉的有三种：METAPOST, PSTricks, PGF。限于篇幅和作者知识面，本文只对这三种工具作简单介绍。METAPOST, PSTricks, PGF 的主要特点如下：

- 工作方式：METAPOST离线绘图，生成的 MPS (一种特殊的 EPS)；PSTricks 和 PGF 都采用在线绘图的方式，也就是在 LaTeX 文档内直接使用绘图命令。
- 兼容性：METAPOST生成的 MPS 需要先转为 PDF 才能被 `pdflatex`使

用；PSTricks 生成的 EPS 和 `pdflatex`不兼容；PGF 提供针对各种驱动的接口，兼容性最好。

- 功能：PSTricks 有 PostScript 作后盾，功能最强；METAPOST擅长处理数学内容；PGF 的流程图有独到之处。

后起之秀 Asymptote 颇有独到之处，英文读者可以参考其网站，中文读者可以参考 milksea 等人编写的文档。

除了上述编程类工具，用户也可以考虑一些面向 LaTeX 的绘图前端，比如 Dia 和 Ipe，或者一些更通用的软件，比如 gnuplot 和 Inkscape。

参考文献

[1] 包太雷. *LaTeX Notes*, 2008. URL http://www.ctan.org/tex-archive/info/latex-notes-zh-cn/.

[2] Steven D. Cochran. *The Subfig Package*, 2005. URL http://www.ctan.org/tex-archive/macros/latex/contrib/subfig/.

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

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

第六章　Metapost

前文提到 Knuth 的 METAFONT可以用来设计矢量字体，但是输出的是点阵格式 PK。1980 年代末 John D. Hobby [1] 设计了一种绘图语言及其编译器，也就是 METAPOST，它从 METAFONT那里获得了大量灵感和源代码。

METAPOST青出于蓝，它输出的是 EPS，而且支持彩色；METAFONT只支持黑白，因为它是用来设计字体的。METAPOST可以在图形上加文字标注，甚至插入 TEX 源码。同时它也从 METAFONT那里继承了一些缺点：数值变量精度较低，且绝对值不能超过 4096；只支持部分 PostScript 功能。雷人可以考虑用 Asymptote 取代 METAPOST。

1994 年之后，Taco Hoekwater [2] 和 Hagen 等人接管了 METAPOST的开发和维护工作。若想深入了解，可以参阅其用户手册[1]。

6.1　准备工作

METAPOST的缺省长度单位是 bp，我们也可以使用 表 2.6 中的其它单位。也可以定义一个缩放系数，把坐标都转换为它的倍数，以后想缩放图形时只要修改这个系数即可。注意变量赋值符号是 :=，而 = 则用于方程式。一个变量在同一源文件中只须定义一次，在其后的图形中都可以使用。

一个 METAPOST源文件 (.mp) 可以包含多个图形，如 例 6.1 所示。代码中每行语句以分号结尾，注释以百分号起始。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。

我们可以用命令行程序 mpost编译 METAPOST源文件，生成一种特殊的 EPS，也就是 MPS；然后再把 MPS 插入 LATEX 源文件中使用。假设源文

[1] 1985 年斯坦福电脑博士，师从 Knuth。后加入贝尔实验室。

[2] 曾学习艺术史和哲学，1992 年辍学加入荷兰军队。退伍后加入克鲁沃学术出版公司 (Kluwer Academic Publishers)，2000 年跳槽到一家电脑公司精灵 (Elvenkind)。LuaTeX 和 ConTEXt 的开发者之一。

```
u := 10pt;    %缩放系数
beginfig(1); %图形起始
...           %绘图命令
endfig;       %图形结尾

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

例 6.1: METAPOST源文件

件名是 `fig.mp`，可以执行以下编译命令，

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

编译后会得到“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中的编号。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。

数字文件名后缀不便于管理，METAPOST为此提供了一个变量来设置输出文件名。我们可以把下面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、···”；或者在在每个图形前面声明各自的名字。

```
outputtemplate "%j-%2c.mps";    %加在源文件头部
outputtemplate "flowchart.mps" %加在每个图形前面
```

`xelatex` 不认识 MPS 格式，所以我们需要使用 `.eps` 后缀，或者用 `\DeclareGraphicsRule` 命令把 `.mps` 声明为 EPS。

```
\DeclareGraphicsRule{.mps}{eps}{.mps}{}
```

6.2 基本图形对象

6.2.1 点和直线

METAPOST的缺省画笔直径为 0.5bp，拿它画点，直径就是 0.5bp；拿它画线，宽度也是 0.5bp。我们可以用 `withpen` 选项为单个绘图命令设置画笔，也可以用 `pickup` 命令为之后所有的绘图命令设置画笔。

在 例 6.2 中，`draw` 命令把几个点以直线段连接起来。`drawdot` 命令在指定坐标画点，为了使它醒目些我们可以换了支粗一点的画笔。

```
filenametemplate "line.eps";
beginfig(1);
draw (0,0)--(4u,0)--(2u,2u)--(0,0) withpen pencircle
    scaled .8pt;
pickup pencircle scaled .8pt;
draw (5u,0)--(9u,0)--(7u,2u)--cycle;
pickup pencircle scaled 3pt;
drawdot (10u,0);
drawdot (14u,0);
drawdot (12u,2u);
endfig;
```

例 6.2: METAPOST点和直线

几段直线或曲线可以构成一条路径 (path)，在路径末尾加个 `cycle` 命令就构成封闭路径 (closed path)。上例中两个三角形看起来都是封闭的，其实后面的才是真正的封闭路径。

6.2.2 预定义形状

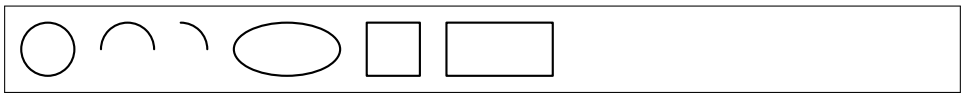
`fullcircle` 命令以原点为圆心画一个单位圆，类似的预定义形状还有 `halfcircle`、`quartercircle`、`unitsquare` 等。注意单位正方形的参考点在左下而不在其中心。在 例 6.3 中，我们使用了不同方向的缩放系数 `xscaled` 和 `yscaled`，把把圆形和正方形变为椭圆和长方形。

6.2.3 曲线

如果把画直线时坐标点之间的 `--` 换成 `..`，我们就得到一条曲线。直线和曲线共用一些点时，它们也可以混在一条语句里画。

METAPOST的曲线用三次贝塞尔算法实现，我们还可以在曲线上使用方向 (direction)、张力 (tension) 和曲率 (curl) 等控制。例 6.4 中左列三图未加任何控制 (代码第四、七、十行)，下排后两图使用了方向控制 (代码第五、六行)，中排后两图使用了张力 (代码第八、九行)，上排后两图使用了曲率 (代码第 11、12 行)。

```
filenametemplate "predefined.eps";
beginfig(3);
pickup pencircle scaled .8pt;
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);
endfig;
```



例 6.3: METAPOST预定义形状

6.3 图形控制

6.3.1 线型

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。

6.3.2 箭头

箭头画法如下，注意反向箭头需要把两个坐标用一对圆括号括起来。

6.3.3 彩色和填充

METAPOST不能使用 `xcolor` 宏包，它只支持 rgb 和 cmyk 色彩模式，预定义了黑、白、红、绿、蓝等颜色。自定义颜色的方法如下，

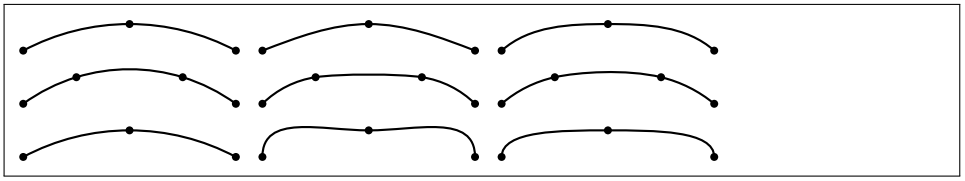
```
color c[];
c1 := .9red + .6green + .3blue;
c2 := (.9,.6,.3);
```

绘图命令一般可以通过 `withcolor` 选项来使用各种颜色。

封闭路径可以用 `fill` 命令来填充。例 6.8 中第三、四行代码定义了一个 `path` 变量，以便后面重用。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

```
filenametemplate "curve.eps";
beginfig(3);
pickup pencircle scaled .8pt;
draw (0,0)..(4u,1u)..(8u,0);
draw (9u,0){up}..(13u,1u){right}..(17u,0){down};
draw (18u,0){up}...(22u,1u){right}...(26u,0){down};
draw (0,2u)..(2u,3u)..(6u,3u)..(8u,2u);
draw (9u,2u)..(11u,3u)..tension 1.5..(15u,3u)..(17u,2u);
draw (18u,2u)..(20u,3u)..tension 1.5 and 1..(24u,3u)..(26u
    ,2u);
draw (0,4u)..(4u,5u)..(8u,4u);
draw (9u,4u){curl 0}..(13u,5u)..{curl 0}(17u,4u);
draw (18u,4u){curl 100}..(22u,5u)..{curl 100}(26u,4u);
pickup pencircle scaled 3pt;
drawdot (0,0); drawdot (4u,1u); drawdot (8u,0);
drawdot (9u,0); drawdot (13u,1u); drawdot (17u,0);
drawdot (18u,0); drawdot (22u,1u); drawdot (26u,0);
drawdot (0,2u); drawdot (2u,3u); drawdot (6u,3u); drawdot
    (8u,2u);
drawdot (9u,2u); drawdot (11u,3u); drawdot (15u,3u);
    drawdot (17u,2u);
drawdot (18u,2u); drawdot (20u,3u); drawdot (24u,3u);
    drawdot (26u,2u);
drawdot (0,4u); drawdot (4u,5u); drawdot (8u,4u);
drawdot (9u,4u); drawdot (13u,5u); drawdot (17u,4u);
drawdot (18u,4u); drawdot (22u,5u); drawdot (26u,4u);
```



例 6.4: METAPOST曲线

`drawoption`，使得其后的绘图命令都使用某种颜色。

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

6.4 图形变换

除了前文提到的缩放，我们还可以对路径进行平移 (`shifted`)、旋转 (`rotated`)、定点旋转 (`rotatedaround`)、镜像 (`reflectedabout`)、倾斜 (`slanted`) 等变换操作。平移的参数是移到的坐标点；旋转的参数是角度，旋转中心是原点；定点旋转的参数是旋转中心；倾斜的参数是倾斜比；镜

```
filenametemplate "dashed.eps";
beginfig(4);
pickup pencircle scaled .8pt;
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;
endfig;
```

例 6.5: METAPOST线型

```
filenametemplate "arrow.eps";
beginfig(5);
pickup pencircle scaled .8pt;
drawarrow (0,0)--(10u,0);
drawarrow reverse ((0,1u)--(10u,1u));
drawdblarrow (0,2u)--(10u,2u);
endfig;
```

例 6.6: METAPOST箭头

像的参数是两点确定的一条直线。

这些变换操作可以任意结合使用。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。例 6.9 中重用了 例 6.8 中定义的路径。

6.5 标注

`label` 命令可以在指定位置加文字标注，该命令有八种后缀，对应着指定坐标点的八个方位 (见 表 6.1) 。`dotlabel` 命令在加标注同时画了个点，它也用同样的方法表示标注的方位。

表 6.1: `label` 命令的方位

top	上	bottom	下	lft	左	rt	右
ulft	左上	urt	右上	llft	左下	lrt	右下

```
filenametemplate "color.eps";
beginfig(6);
pickup pencircle scaled .8pt;
draw (0,0)--(10u,0) withcolor red;
draw (0,1u)--(10u,1u) withcolor green;
draw (0,2u)--(10u,2u) withcolor blue;
endfig;
```

例 6.7: METAPOST彩色

```
filenametemplate "fill.eps";
beginfig(7)
path p;
p := (0,0)--(2,0)--(1,1.732)--cycle;
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;
endfig;
```

例 6.8: METAPOST填充

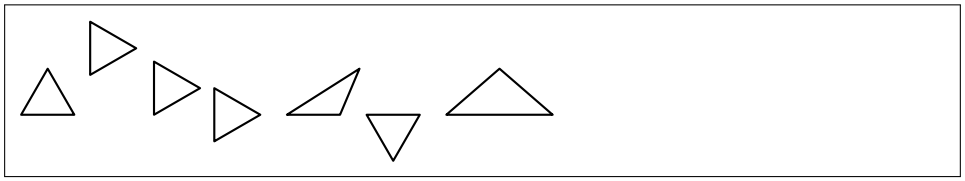
我们也可以用一对 `btex` 和 `etex` 来嵌入一些 TEX 内容，比如数学标注 (例 6.10 代码第 14、18、19 行) 。`mpost` 会把 TEX 内容存到一个临时文件，调用 `tex` 编译它生成 DVI；`mpost` 把 DVI 转换为 METAPOST内容存到一个 `.mpx` 文件中，然后再把它嵌入输出的 MPS。

METAPOST中也可以嵌入复杂的 LATEX 代码，比如字体和语言等的设置。这时需要在源文件头尾加两对 `verbatimtex` 和 `etex` 命令，分别包含前置和后置处理代码，原图形代码放在这两对命令之间。可惜 METAPOST不支持嵌入 XELATEX 代码，所以其文字标注不能使用后者的字体功能。

MPS 缺省不嵌入字体，当它包含文字时，GSview 就不能正常查看；但是把这种 MPS 插入文档生成的 PDF 还是正常的，因为驱动会自行处理字体。我们可以强制 MPS 嵌入字体，一种方法是在源文件头部加一行语句 (例 6.10 代码第一行) ，另一种方法是在编译时加一个参数，

```
mpost \prologues:=3; input fig.mp
```

```
filenametemplate "transform.eps";
beginfig(8);
pickup pencircle scaled .8pt;
draw p scaled u;
draw p scaled u shifted (3u,0) rotated 30;
draw p scaled u rotated 30 shifted (5u,0);
draw p scaled u rotatedaround ((2u,0),30) shifted (7u,0) ;
draw p scaled u slanted 1 shifted (10u,0);
draw p scaled u reflectedabout ((0,0),(2u,0)) shifted (13u
   ,0);
draw p xscaled 2u yscaled u shifted (16u,0);
endfig;
```



例 6.9: METAPOST图形变换

6.6 编程

6.6.1 数据类型和变量

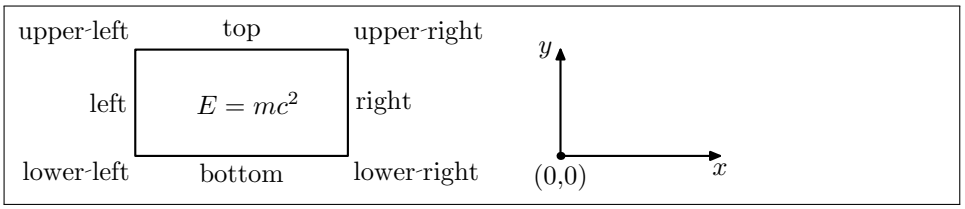
METAPOST中有十种基本数据类型: `numeric`、`pair`、`path`、`pen`、`color`、`cmykcolor`、`transform`、`string`、`boolean`、`picture`。我们已经接触过其中几种，比如缩放系数`u`是`numeric`，点的坐标是`pair`，几个点用直线连起来是一个`path`，`pencircle`是一种`pen`，红、绿、蓝都是`color`，`scaled`、`shifted`、`rotated`都是`transform`。

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
```

```
prologues:=3;
filenametemplate "label.eps";
beginfig(9);
pickup pencircle scaled .8pt;
draw unitsquare xscaled 8u yscaled 4u;
label.top ("top", (4u,4u));
label.bot ("bottom", (4u,0));
label.lft ("left", (0,2u));
label.rt ("right", (8u,2u));
label.ulft ("upper left", (0,4u));
label.urt ("upper right", (8u,4u));
label.llft ("lower left", (0,0));
label.lrt ("lower right", (8u,0));
label.rt (btex $E=mc^2$ etex, (2u,2u));
drawarrow (16u,0)--(22u,0);
drawarrow (16u,0)--(16u,4u);
dotlabel.bot ("(0,0)", (16u,0));
label.bot (btex $x$ etex, (22u,0));
label.lft (btex $y$ etex, (16u,4u));
endfig;
```



例 6.10: METAPOST标注

6.6.2 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 6.2 列出一些常用数学函数。

6.6.3 循环

当执行重复任务时，循环语句可以让程序变得简洁 (见 例 6.11) 。

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` 的简写方式。

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

表 6.2: METAPOST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

```
filenametemplate "loop.eps";
beginfig(10);
pickup pencircle scaled .8pt;
drawarrow (0,0)--(10u,0);
drawarrow (0,0)--(0,4u);
draw (0,0) %注意这里没有分号
for x=1 upto 3: ..(x*x,x)*u endfor;
endfig;
```

例 6.11: METAPOST循环

参考文献

[1] John D. Hobby, Taco Hoekwater, and Hans Hagen. *MetaPost: A User's Manual*, 2010. URL http://www.tug.org/metapost/.

第七章　PSTricks

PSTricks 是一个基于 PostScript 的宏包，有了它用户就可以直接在 LaTeX 文档中插入绘图命令。PSTricks 早期的作者是 van Zandt，初始开发年月不详，1997 年之后 Denis Girou [1] 和 Herbert Voß [2] 接管了维护工作。

表 7.1 列出了一些可以和 PSTricks 配合使用的辅助宏包。

表 7.1: PSTricks 辅助宏包

multido	循环	pst-3dplot	三维绘图
pst-node	示意图	pst-solides3d	三维实体
pst-tree	树状图	pst-circ	电路
pst-plot	函数绘图	pst-labo	化学
pst-func	数学函数	pst-geo	地理
pst-eucl	几何函数	pstricks-add	杂项

7.1　准备工作

PSTricks 中缺省长度单位是 1cm，我们也可以设置自己的单位。绘图命令一般要放在 `pspicture` 环境里，其参数是一个矩形的左下角和右上角，如果左下从原点开始可以省略该点坐标。这样 LaTeX 就会给它预留空间，注意这个矩形要能容纳所有图形对象。

```
\psset{unit=10pt}
\begin{pspicture}(0,0)(4,2)
...
\end{pspicture}
```

[1] 供职于法国国家科学研究中心 (National Centre for Scientific Research, CNSR)。

[2] 电力工程博士，高中数学、物理、电脑教师，柏林自由大学 (Free University of Berlin) 讲师。LaTeX 3 项目成员，十几本书的作者。

另外需要注意的是，嵌入 LaTeX 的 PSTricks 生成的是 PostScript，dvips 认得自家人，xdvipdfmx 也可以处理。dvipdfm 和 pdflatex 则不能；如果使用后两种驱动，需要先行生成 EPS 或 PDF。

pst-eps 宏包能够在线处理 PSTricks 代码并生成 EPS，这样用户就可以在同一 LaTeX 源文件中插入该 EPS。然而 dvipdfmx 不能正确处理 pst-eps 生成的 EPS，它和 \rput、\uput、\psaxes 等命令都不兼容。

```
%fig.tex
\documentclass{article}
\usepackage{pst-pdf}
\begin{document}
\begin{pspicture}(0,0)(4,2)
...
\end{pspicture}
\end{document}
```

例 7.1: pst-pdf 宏包

较好的方法是用 pst-pdf 宏包生成包含 PSTricks 图形的 EPS，然后再根据需要转为 PDF。每个 pspicture 环境中的内容会自成一页，方便插入文档。下面 dvips 命令的 -E 参数指示生成 EPS。

```
latex fig(.tex)
dvips fig(.dvi) -E fig.eps
ps2pdf fig.eps fig.pdf
```

```
\documentclass{article}
\usepackage[active,tightpage,xetex]{preview}
\usepackage{pstricks}
\begin{document}
\begin{preview}
\begin{pspicture}(0,0)(4,2)
...
\end{pspicture}
\end{preview}
\end{document}
```

例 7.2: preview 宏包

使用 dvips 或 xdvipdfmx 时也可以考虑这种生成独立图形的方法，因为直接在 LaTeX 源文件中使用绘图命令，编译时间会比较长。使用 pst-pdf 宏包时的编译命令较繁琐，用两个命令生成 EPS，第三个命令才得到 PDF。

如果使用 xdvipdfmx，可以改用 preview 宏包 (见 例 7.2)，这样直接生成 PDF，而且文件体积较小。每个 preview 环境中的内容也会自成一页。

7.2 基本图形对象

7.2.1 点和直线

\dot 命令画一个点，\dots 命令画多个点。因为点是有直径的，pspicture 的尺寸需要稍大一点。\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，也可以设置为圆角。

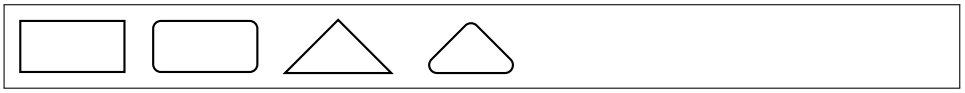
```
\begin{pspicture}(-.2,-.2)(14,2.2)
\psdot(0,0)
\psdots(4,0)(2,2)
\psline(5,0)(7,2)(9,0)
\psline[linearc=.3](10,0)(12,2)(14,0)
\end{pspicture}
```

例 7.3: PStricks 点和直线

7.2.2 矩形和多边形

矩形用 \psframe 命令，其参数就是矩形左下角和右上角的坐标。多边形用 \pspolygon 命令，语法和 \psline 类似，但是它会形成封闭路径。矩形和多边形都可以设置圆角。

```
\begin{pspicture}(19,3)
\psframe(0,0)(4,3)
\psframe[framearc=.3](5,0)(9,3)
\pspolygon(10,0)(14,0)(12,3)
\pspolygon[linearc=.3](15,0)(19,0)(17,3)
\end{pspicture}
```

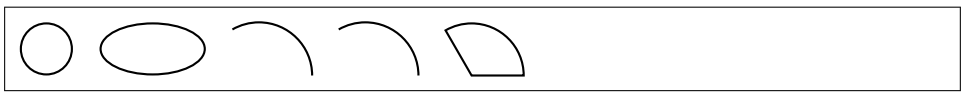


例 7.4: PStricks 矩形和多边形

7.2.3 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。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。

```
\begin{pspicture}(19,2)
\pscircle(1,1){1}
\psellipse(5,1)(2,1)
\psarc(9,0){2}{0}{120}
\psarcn(13,0){2}{120}{0}
\pswedge(17,0){2}{0}{120}
\end{pspicture}
```

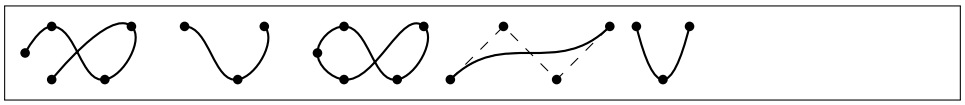


例 7.5: PStricks 圆、椭圆、圆弧、扇形

7.2.4 曲线

\pscurve 命令把一系列点用平滑曲线连接起来；\psecurve 命令不显示曲线的两个端点；\psccurve 命令则把曲线封闭起来。showpoints 参数用来指示是否显示曲线的构成点，它也可用于其它绘图命令。

```
\begin{pspicture}(-0.2,-0.2)(25.2,2.2)
\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[showpoints=true](16,0)(18,2)(20,0)(22,2)
\psparabola[showpoints=true](25,2)(24,0)
\end{pspicture}
```



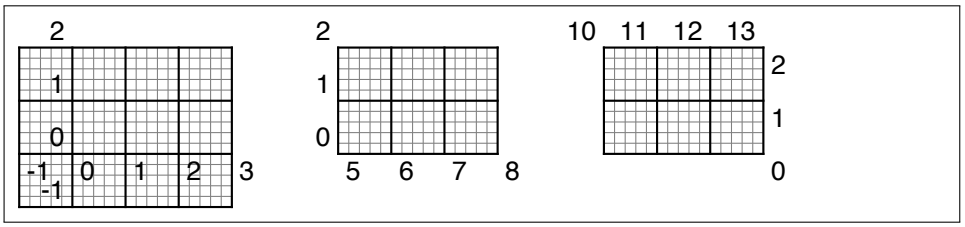
例 7.6: PStricks 曲线

贝塞尔曲线用 \psbezier 命令，其参数就是曲线的控制点。抛物线用 \psparabola 命令，它有两个参数，一个是抛物线通过的某点，另一个是抛物线的顶点。

7.2.5 网格和坐标轴

科技制图通常会用到坐标和网格。`\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。

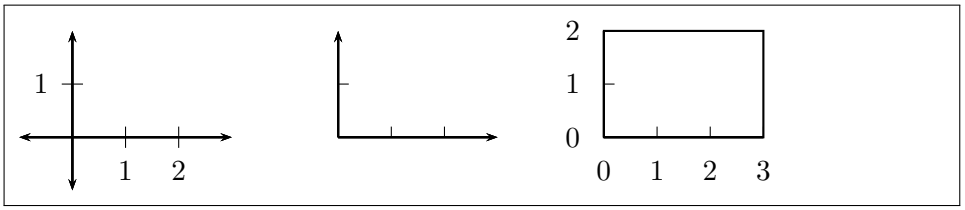
```
\psset{unit=20pt}
\begin{pspicture}(-1,-1)(13.5,2.5)
\psgrid(0,0)(-1,-1)(3,2)
\psgrid(5,0)(8,2)
\psgrid(13,2)(10,0)
\end{pspicture}
```



例 7.7: PStricks 网格

坐标轴可以用 `pst-plot` 宏包的 `\psaxes` 命令。它的参数和 `\psgrid` 的类似，刻度和标注的设置都很方便，也可以把坐标轴改成矩形框。

```
\psset{unit=20pt}
\begin{pspicture}(-1,-1)(13.4,2.4)
\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)
\end{pspicture}
```



例 7.8: PStricks 坐标轴

7.3 图形控制

7.3.1 线宽和线型

PSTricks 中缺省线条是 0.8pt 的实线。线宽和线型参数使用方法如下，

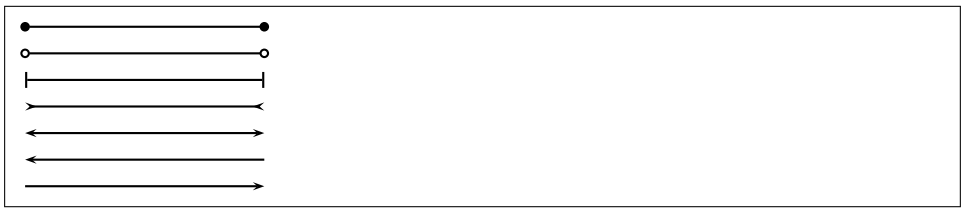
```
\begin{pspicture}(0,-0.1)(9,2.1)
\psline[linewidth=1.5pt](0,0)(9,0)
\psline[linestyle=dotted](0,1)(9,1)
\psline[linestyle=dashed](0,2)(9,2)
\end{pspicture}
```

例 7.9: PStricks 线宽和线型

7.3.2 箭头

以下参数可以控制绘图命令的箭头。

```
\begin{pspicture}(-0.2,-0.2)(9.2,6.2)
\psline{->}(0,0)(9,0)
\psline{<-}(0,1)(9,1)
\psline{<->}(0,2)(9,2)
\psline{>-<}(0,3)(9,3)
\psline{|-|}(0,4)(9,4)
\psline{o-o}(0,5)(9,5)
\psline{*-*}(0,6)(9,6)
\end{pspicture}
```



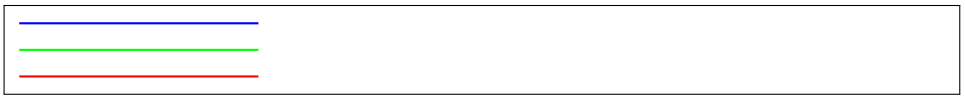
例 7.10: PStricks 箭头

7.3.3 颜色和填充

PSTricks 预定义了 black, darkgray, gray, lightgray, white 等灰度颜色，和 red, green, blue, cyan, magenta, yellow 等彩色。它可以使用 `xcolor` 宏包。设置线

条颜色的方法如 例 7.11 所示，

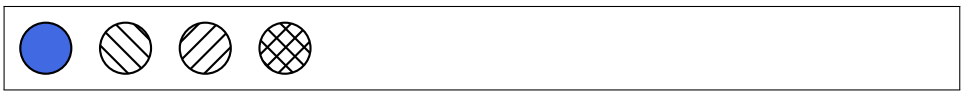
```
\begin{pspicture}(0,-0.1)(9,2.1)
\psline[linecolor=red](0,0)(9,0)
\psline[linecolor=green](0,1)(9,1)
\psline[linecolor=blue](0,2)(9,2)
\end{pspicture}
```



例 7.11: PStricks 彩色

设置填充模式和填充颜色的方法如 例 7.12。注意只有封闭路径才可以填充。

```
\begin{pspicture}(11,2)
\pscircle[fillstyle=solid,fillcolor=RoyalBlue](1,1){1}
\pscircle[fillstyle=vlines](4,1){1}
\pscircle[fillstyle=hlines](7,1){1}
\pscircle[fillstyle=crosshatch](10,1){1}
\end{pspicture}
```



例 7.12: PStricks 填充

7.3.4 全局设置

前文提到过用 \pseset 命令设置长度单位，它还可以用来设置线宽、线型、颜色等全局参数。

```
\psset{linewidth=1pt, linestyle=dashed, linecolor=Silver,
   fillcolor=Lavender, fillstyle=crosshatch}
```

7.4 图形变换

origin 参数让一个图形对象平移到指定的位置。\rput 命令可以对一个图形对象同时进行旋转和平移操作。它有四个参数：

语法：\rput[参考点]{旋转角度}{平移坐标}{操作对象}

1. 参考点，可选。其取值见 表 7.2，缺省是左下。
2. 旋转角度，可以取任意角度，也可以用 U、L、D、R 分别代表 0°、90°、180°、270° 。
3. 平移到的位置。
4. 操作对象。

表 7.2: \rput 命令的参考点

水平方向		垂直方向	
l	左	t	上
r	右	b	下

```
\begin{pspicture}(12,3.2)
\psframe(0,0)(3,2)
\psframe[origin={4,0}](0,0)(3,2)
\rput{30}(9,0){\psframe(0,0)(3,2)}
\end{pspicture}
```

例 7.13: PStricks 平移和旋转

如果要进行全局坐标变换，可以考虑使用 translate, scale, rotate, swapaxes 等命令。

7.5 标注

\rput 命令还可以在指定的坐标点标注文字，例 7.14 中的参考点参数指示把文字分别标注在某点的左边、右边、上边。它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，取值可以是角度或字母 (见 表 7.3) 。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，比较容易混淆。

若想深入了解 PSTricks，可以参阅其用户手册[1]。

```
\begin{pspicture}(-0.8,-0.4)(12.3,3.3)
\pspolygon(0,0)(4,0)(2,2)
\rput[r](0,0){A}
\rput[l](4,0){B}
\rput[b](2,2){C}
\pspolygon(7,0)(11,0)(9,2)
\uput[l](7,0){A}
\uput[r](11,0){B}
\uput[u](9,2){C}
\end{pspicture}
```

例 7.14: PStricks 标注

表 7.3: \uput 命令的角度参数

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

参考文献

[1] Timothy van Zandt. *PSTricks User's Guide*, 2007. URL http://www.tug.org/PSTricks/.

广告位招租

第八章　PGF

PGF 和 Beamer 的作者都是 Till Tantau (1975–) [1]。Tantau 当初开发 Beamer 是为了准备 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[1]。

8.1　准备工作

一般人们并不直接使用 PGF 底层命令，而是通过它前端 TikZ 来调用。在引用 `tikz` 宏包之前，用户需要设置 PGF 系统驱动。比如 `dvipdfmx` 的设置方法如下，使用 `pdflatex` 和 `xelatex` 时，它知道驱动是谁。

```
\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}}
```

TikZ 提供 `\tikz` 命令和 `tikzpicture` 环境，具体绘图指令可以放在 `\tikz` 后面，也可以放在 `tikzpicture` 中间。两者效果相同，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}
```

[1] 柏林工业大学1999 年电脑学士，2001 年数学学士，2003 年电脑博士。2004 年伯克利访问学者，2005 年吕贝克大学 (University of Lübeck) 理论计算机研究所教授。

7.1 小节提到，为了节省编译时间，我们可以用 `preview` 宏包生成独立的图形文件。对于 PGF 我们也可以如法炮制，虽然这样做不是必须的。

```
\documentclass{article}
\usepackage[active,tightpage,xetex]{preview}
\usepackage{tikz}

\begin{document}
\begin{preview}
\begin{tikzpicture}
...
\end{tikzpicture}
\end{preview}
\end{document}
```

例 8.1: 制作独立的 PGF 图形文件

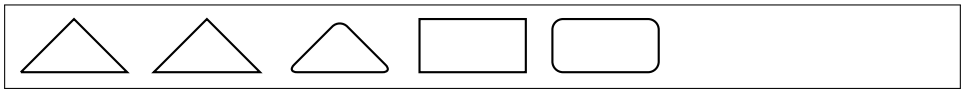
8.2 基本图形对象

8.2.1 直线和矩形

PGF 绘图命令的语法和 METAPOST有点类似。在 例 8.2 中，`\draw` 称为一个命令，它后面的 `--`、`cycle`、`rectangle` 等称为操作，`[rounded corners]` 称作一个选项。`--` 用来画直线，`rectangle` 画矩形；`cycle` 用来封闭路径，前两个三角形看起来一样，其实只有第二个才是真正的封闭路径；`[rounded corners]` 用来给图形加圆角。

操作都需要一个起始点参数，比如第一行代码第一个 `--` 操作的起始点是 (0,0)；第二行 `cycle` 的起始点是 (7,2)；第四行 `rectangle` 的起始点是 (15,0)，也就是矩形的一个顶点，(19,2) 是其对角顶点。

```
\draw (0,0)--(4,0)--(2,2)--(0,0);
\draw (5,0)--(9,0)--(7,2)--cycle;
\draw [rounded corners] (10,0)--(14,0)--(12,2)--cycle;
\draw (15,0) rectangle (19,2);
\draw [rounded corners] (20,0) rectangle (24,2);
```



例 8.2: PGF 直线和矩形

8.2.2 圆、椭圆、弧

圆、椭圆、弧等形状的画法如下。圆的参数是圆心和半径，椭圆的参数是中心、长径、短径。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 (1,1) circle (1);
\draw (5,1) ellipse (2 and 1);
\draw (10,1) arc (0:270:1);
\draw (15,1) arc (0:270:2 and 1);
```

例 8.3: PGF 圆、椭圆、弧

8.2.3 曲线

我们把直线的 -- 换成 ..，就得到贝塞尔曲线，它需要至少一个控制点(例 8.4 第一行)。抛物线用 parabola，代码第六行的 (5,1) 是它的起始点，(7.414,2) 是终止点，bend (6,0)指定了顶点。

```
\draw (0,0)..controls (2,2) and (4,2)..(4,0);
\filldraw (0,0) circle (.1)
    (2,2) circle (.1)
    (4,2) circle (.1)
    (4,0) circle (.1);
\draw (5,1) parabola bend (6,0) (7.414,2);
\filldraw (5,1) circle (.1)
    (6,0) circle (.1)
    (7.414,2) circle (.1);
\draw (8,0) sin (10,2) cos (12,0);
\filldraw (8,0) circle(.1)
    (10,2) circle(.1)
    (12,0) circle(.1);
```

例 8.4: PGF 曲线

正弦和余弦都需要起、止点，第 11 行代码中的余弦操作看起来少一个起始点，其实它是接着正弦的末端画的。\filldraw 命令是为了标明曲线上的点，这方面 PSTricks 比 METAPOST和 PGF 都方便，一个 showpoints 参

数就都搞定了。

8.2.4 网格

网格的画法如下，其缺省步长是 1cm。`grid` 操作需要起止点两个参数。`help lines` 参数指示用 0.2pt 的灰线。

```
\draw [step=5pt] (0,0) grid (3,2);
\draw [help lines,step=5pt] (4,0) grid (7,2);
```

例 8.5: PGF 网格

8.3 图形控制

8.3.1 箭头

各种箭头的画法如下:

```
\draw [->] (0,0)--(9,0);
\draw [<-] (0,1)--(9,1);
\draw [<->] (0,2)--(9,2);
\draw [>->>] (0,3)--(9,3);
\draw [|<->|] (0,4)--(9,4);
```

例 8.6: PGF 箭头

8.3.2 线宽和线型

PGF 中线条的缺省宽度是 0.4pt，线型是实线。改变线宽和线型的方法见 例 8.7。

```
\draw [line width=2pt] (0,0)--(9,0);
\draw [dotted] (0,1)--(9,1);
\draw [densely dotted] (0,2)--(9,2);
\draw [loosely dotted] (0,3)--(9,3);
\draw [dashed] (0,4)--(9,4);
\draw [densely dashed] (0,5)--(9,5);
\draw [loosely dashed] (0,6)--(9,6);
```

例 8.7: PGF 线宽和线型

8.3.3 颜色和填充

PGF 可以使用 `xcolor` 宏包的色彩功能。颜色和填充的用法见 例 8.8，其中 `\filldraw` 命令可以用不同颜色画线和填充。注意封闭路径才可以填充。PGF 还有十几种填充模式，可惜 XeTeX 不支持。

```
\draw[Red] (0,0)--(9,0);
\draw[Green] (0,1)--(9,1);
\draw[Blue] (0,2)--(9,2);
\fill[Wheat] (11,1) circle (1);
\filldraw[draw=Silver, fill=Lavender] (14,1) circle (1);
```

例 8.8: PGF 颜色和填充

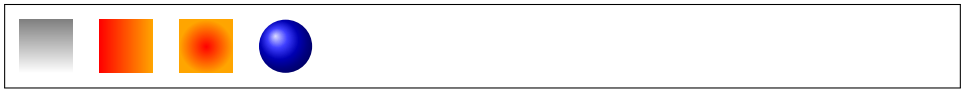
8.3.4 渐变和阴影

`\shade` 命令可以产生渐变和阴影效果，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（例 8.9）。

8.3.5 样式

PGF 比 METAPOST和 PSTricks 多了一个有趣的概念：样式 (style)，它像面向对象的类一样可以继承，语法和 HTML 的 CSS 相近。在 例 8.10 中我

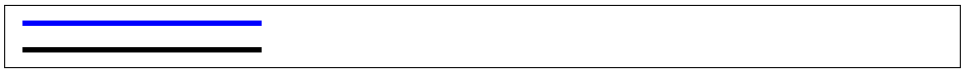
```
\shade (0,0) rectangle (2,2);
\shade[left color=Red,right color=Orange] (3,0) rectangle
    (5,2);
\shade[inner color=Red,outer color=Orange] (6,0) rectangle
     (8,2);
\shade[ball color=Blue] (10,1) circle (1);
```



例 8.9: PGF 阴影

们先定义了两种样式，然后就可以在绘图命令中使用它们，

```
\tikzset{
    myline/.style={line width=2pt},
    myblueline/.style={myline,Blue}
}
\draw[myline] (0,0)--(9,0);
\draw[myblueline] (0,1)--(9,1);
```



例 8.10: PGF 全局样式

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。前者是全局有效，后者则是局部范围有效。

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

例 8.11: PGF 局部样式

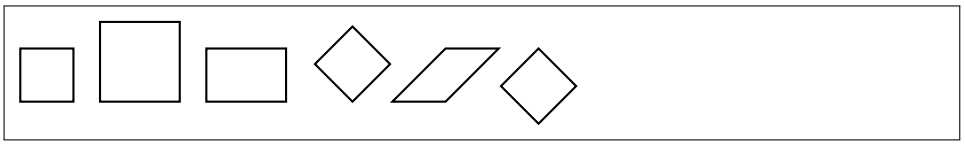
注意在样式中预定义长度单位有时会失效，所以最好使用绝对单位。

8.4 图形变换

对图形对象可以进行缩放 (scale)、平移 (shift)、倾斜 (slant)、旋转 (rotate)、定点旋转 (rotate around) 等变换操作。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单向平移选项 (xshift 或 yshift) 失效。

```
\draw (0,0) rectangle (2,2);
\draw[shift={(3,0)},scale=1.5] (0,0) rectangle (2,2);
\draw[xshift=70pt,xscale=1.5] (0,0) rectangle (2,2);
\draw[xshift=125pt,rotate=45] (0,0) rectangle (2,2);
\draw[xshift=140pt,xslant=1] (0,0) rectangle (2,2);
\draw[xshift=175pt,rotate around={45:(2,2)}] (0,0)
   rectangle (2,2);
```



例 8.12: PGF 图形变换

8.5 示意图

8.5.1 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 box 样式，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。

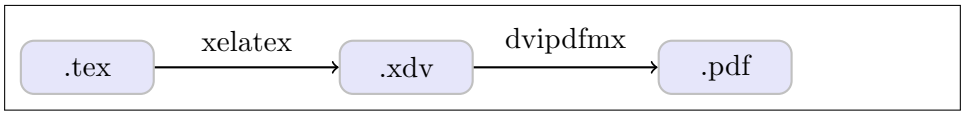
```
\tikzset{
    box/.style={rectangle,rounded corners=5pt,
    minimum width=50pt,minimum height=20pt,inner sep=5pt,
    draw=Silver,fill=Lavender}
}
```

例 8.13: PGF box 样式

8.5.2 流程图

除了上述属性，节点还可以有名字、位置等属性。在 例 8.12 中，我们先画了三个有名字和边框的节点，也就是文本框；然后用两跳箭头连线把文本框连接起来，注意连接时要引用文本框的名字；接着在连线上加了两个没有名字和边框的标签。

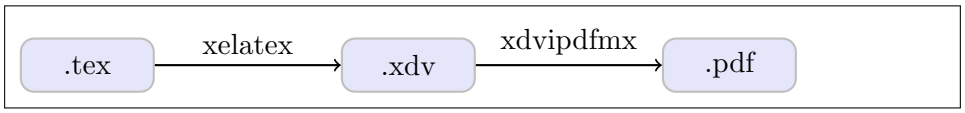
```
\node[box] (tex) at(0,0) {.tex};
\node[box] (xdv) at(12,0) {.xdv};
\node[box] (pdf) at(24,0) {.pdf};
\draw[->] (tex)--(xdv);
\draw[->] (xdv)--(pdf);
\node at (6,1) {xelatex};
\node at (18,1) {dvipdfmx};
```



例 8.14: PGF 流程图

例 8.14 中的节点都使用了绝对位置，我们还可以使用更灵活一点的相对位置。在 例 8.15 中，我们把 xdv 节点设在在 tex 节点右边 70pt 处 (前面定义的基本长度单位是 10pt)，而 pdf 节点又在 xdv 节点右边 70pt 处。

```
\node[box] (tex) {.tex};
\node[box,right=7 of tex] (xdv) {.xdv};
\node[box,right=7 of xdv] (pdf) {.pdf};
\path (tex) edge[->]  node[auto] {xelatex} (xdv)
    (xdv) edge[->] node[auto] {xdvipdfmx} (pdf);
```



例 8.15: PGF 又一个流程图

节点间的连线换为专门用来连接节点的 `edge`；标签也改成相对位置，自动排列在 `edge` 上方。

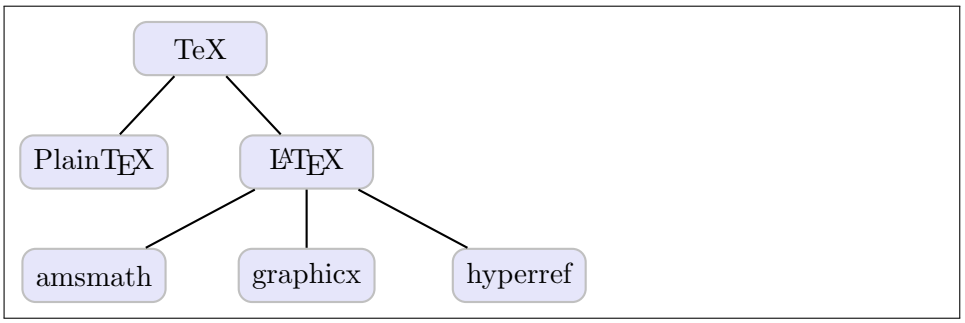
8.5.3 树

例 8.16 是一棵简单的树。`child` 关键字用来声明子节点；`sibling distance` 选项可以控制相邻节点之间的距离，预定义长度单位对此选项失效。

8.5.4 预定义节点形状

PGF 中节点的基本形状只有矩形和圆形，`shapes.geometric` 库预定义了其他一些形状，比如菱形 (diamond)、梯形 (trapezium)、半圆 (semicircle)

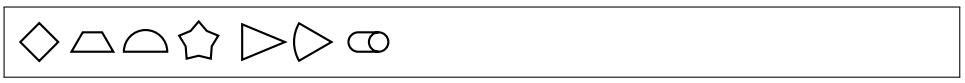
```
\begin{tikzpicture}[sibling distance=80pt]
\node[box] {TeX}
    child {node[box] {Plain\TeX}}
    child {node[box] {\LaTeX}
        child {node[box] {amsmath}}
        child {node[box] {graphicx}}
        child {node[box] {hyperref}}
    };
\end{tikzpicture}
```



例 8.16: PGF 好大一棵树

、星形 (star)、等腰三角形 (isosceles triangle)、扇形 (circular sector)、圆柱 (cylinder)、正多边形 (regular polygon) 等形状。如要使用某库的功能，需要先引用它，其引用方法类似于宏包的引用，见 例 8.17 代码第一行。

```
\usetikzlibrary{shapes.geometric}
\node[diamond,draw] at(0,0) {};
\node[trapezium,draw] at(2,0) {};
\node[semicircle,draw] at(4,0) {};
\node[star,draw] at(6,0) {};
\node[isosceles triangle,draw] at(8,0) {};
\node[circular sector,draw] at(10,0) {};
\node[cylinder,draw] at(12,0) {};
```



例 8.17: PGF 预定义节点形状

例 8.18 中代码第一行指出每个节点都是正多边形，没有这个声明的话每个节点都要重复拼写 `regular polygon`。

```
\begin{tikzpicture}[every node/.style={regular polygon}]
\node[regular polygon sides=3,draw] at(2,0) {};
\node[regular polygon sides=4,draw] at(4,0) {};
\node[regular polygon sides=5,draw] at(6,0) {};
\node[regular polygon sides=6,draw] at(8,0) {};
\node[regular polygon sides=7,draw] at(10,0) {};
\node[regular polygon sides=8,draw] at(12,0) {};
\end{tikzpicture}
```

例 8.18: PGF 预定义正多边形节点

8.6 编程

8.6.1 循环语句

包老师正准备跟 例 8.18 里那样的冗长代码拼个你死我活，猛然发现 TikZ 提供了循环语句。Only you 能精简我代码，Only you´´´

```
\begin{tikzpicture}[every node/.style={regular polygon}]
\foreach \x in {3,4,5,6,7,8}{
    \node[regular polygon sides=\x, draw] at(\x*2,0) {};
}
\end{tikzpicture}
```

例 8.19: PGF 循环语句

8.6.2 数据图

PGF 有强大的数据绘图 (plot) 功能，支持多种绘图方法：直接给点，读取外部数据文件，函数绘图，调用 Gnuplot。其中函数绘图可以调用 PGF 数学引擎，进行基本算术和 20 多种函数的运算。

例 8.20 是一个简单的指数函数，其中标注使用了 LaTeX 数学公式，`domain` 选项用来设置值域。

```
\draw[->] (-0.2,0)--(6,0) node[right] {$x$};
\draw[->] (0,-0.2)--(0,6) node[above] {$f(x)$};
\draw[domain=0:4] plot (\x,{0.1*exp(\x)}) node[right] {$f(
    x)=\frac{1}{10}e^x$};
```

例 8.20: PGF 函数图

参考文献

[1] Till Tantau. *TikZ and PGF Manual*, 2008. URL http://www.ctan.org/tex-archive/graphics/pgf/.

广告位招租

第九章　表格

9.1　简单表格

tabular 环境提供了最简单的表格功能。它用 \hline 命令表示横线，| 表示竖线；用 & 来分列，用 \\ 来换行；每列可以采用居中、居左、居右等横向对齐方式，分别用 l、c、r 来表示。

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

操作系统	发行版	编辑器
Windows	MikTeX	TexMakerX
Unix/Linux	teTeX	Kile
Mac OS	MacTeX	TeXShop
通用	TeX Live	TeXworks

例 9.1: 简单表格

在插图一章中我们介绍了一种图形浮动环境 figure；表格也有一种类似的浮动环境 table，其标题和交叉引用的用法和图形浮动环境类似。我们可以用它给 例 9.1 中的表格穿件马甲，顺便把表格简化为科技文献

中常用的三线表。例 9.1 中的三条横线一样粗细，如果嫌它不够美观，可以使用 Simon Fear [1] 的 booktabs 宏包[1]。三条横线就分别用 \toprule、\midrule、\bottomrule 等命令表示。改进后的表格见 例 9.2。

```
\begin{table}[htbp]
\centering
\begin{tabular}{lll}
    \toprule
    操作系统 & 发行版 & 编辑器 \\
    \midrule
    Windows     & MikTeX    & TexMakerX \\
    Unix/Linux  & teTeX     & Kile \\
    Mac OS      & MacTeX    & TeXShop \\
    通用        & TeX Live  & TeXworks \\
    \bottomrule
\end{tabular}
\end{table}
```

操作系统	发行版	编辑器
Windows	MikTeX	TexMakerX
Unix/Linux	teTeX	Kile
Mac OS	MacTeX	TeXShop
通用	TeX Live	TeXworks

例 9.2: 浮动三线表

tabular 环境中的行可以采用居顶、居中、居底等纵向对齐方式，分别用 t、c、b 来表示，缺省的是居中对齐。列之间的分隔符也可以改用其他符号，比如用 || 来画双竖线。

语法：[纵向对齐]{横向对齐和分隔符}

9.2 宽度控制

有时我们需要控制某列的宽度，可以将其对齐方式参数从 l、c、r 改为 p{宽度}。这时纵向对齐方式是居顶，t、c、b等参数失效。

[1]这名字看起来像个笔名。《圣经 · 新约》路加福音卷有一个故事：耶稣于某湖边传教时，命渔人西门撒网，得鱼甚多。西门惊诧，五体投地，口称罪人。耶稣道：西门莫怕，打渔这个职业没什么前途，你还是跟我打人罢 (Jesus said unto Simon, Fear not; from henceforth thou shalt catch men.) 。西门遂忝列夫子门墙。另外有个恐怖小说系列 “Fear Street” 里的主角也叫西门怕怕。

```
\begin{table}[htbp]
\centering
\begin{tabular}{p{80pt}p{80pt}p{80pt}}
    \toprule
    操作系统 & 发行版 & 编辑器 \\
    \midrule
    Windows     & MikTeX    & TexMakerX \\
    Unix/Linux  & teTeX     & Kile \\
    Mac OS      & MacTeX    & TeXShop \\
    通用        & TeX Live  & TeXworks \\
    \bottomrule
\end{tabular}
\end{table}
```

操作系统	发行版	编辑器
Windows	MikTeX	TexMakerX
Unix/Linux	teTeX	Kile
Mac OS	MacTeX	TeXShop
通用	TeX Live	TeXworks

例 9.3: 控制列宽

使用宽度控制参数之后，表格内容缺省居左对齐。我们可以用列前置命令 `>{}` 配合 `\centering`、`\raggedleft` 命令来把横向对齐方式改成居中或居右。列前置命令仅对紧邻其后的一列有效，其语法如下：

语法：`>{命令}列参数`

```
\begin{table}[htbp]
\centering
\begin{tabular}{p{80pt}>{\centering}p{80pt}>{\raggedleft\
    arraybackslash}p{80pt}}
    \toprule
    操作系统 & 发行版 & 编辑器 \\
    \midrule
    Windows     & MikTeX    & TexMakerX \\
    Unix/Linux  & teTeX     & Kile \\
    Mac OS      & MacTeX    & TeXShop \\
    通用        & TeX Live  & TeXworks \\
    \bottomrule
\end{tabular}
\end{table}
```

操作系统	发行版	编辑器
Windows	MikTeX	TexMakerX
Unix/Linux	teTeX	Kile
Mac OS	MacTeX	TeXShop
通用	TeX Live	TeXworks

例 9.4: 控制列宽和横向对齐

若要控制整个表格的宽度，可以使用 Carlisle 的 `tabularx` 宏包[2] 的同名环境，其语法如下，其中 `X` 参数表示某列可以折行。

语法：`{表格宽度}{横向对齐、分隔符、折行}`

```
\begin{table}[htbp]
\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。\\
    \bottomrule
\end{tabularx}
\end{table}
```

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。

例 9.5: 控制表格宽度

如果想把纵向对齐方式改为居中和居底，可以使用 Mittelbach 和 Carlisle 的 `array` 宏包[3]，它提供了另两个对齐方式参数：`m{宽度}`、`b{宽度}`。

9.3 跨行跨列

有时表格某单元格需要横跨几列，我们可以使用 `\multicolumn` 命令，同时使用 `booktabs` 宏包的 `\cmidrule` 命令来画横跨几列的横线。它们的语法如下：

```
\begin{table}[htbp]
\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    & TexMakerX \\
    Unix/Linux  & teTeX     & Kile \\
    Mac OS      & MacTeX    & TeXShop \\
    通用        & TeX Live  & TeXworks \\
    \bottomrule
\end{tabular}
\end{table}
```

	常用工具	
操作系统	发行版	编辑器
Windows	MikTeX	TexMakerX
Unix/Linux	teTeX	Kile
Mac OS	MacTeX	TeXShop
通用	TeX Live	TeXworks

例 9.6: 跨列表格

语法：`\multicolumn{横跨列数}{对齐方式}{内容}`

语法：`\cmidrule{起始列-结束列}`

9.4 数字表格

当表格中包含大量数字时，手工调整小数点和数位的对齐很麻烦，这时可以使用 Rochester 的 `warpcol` 宏包[4]。它为 `tabular` 环境提供了一个列对齐参数 `P`，其语法如下，其中 `m` 和 `n` 分别是小数点前后的位数，数字前的负号可选。

语法：`P{-m.n}`

例 9.7 中使用 `multicolumn` 命令是为了保护表头，防止它们被 `P` 参数误伤。把跨列命令的列数设为 1 是设置单元格格式的一种常用方法。

```
\begin{table}[htbp]
\centering
\begin{tabular}{P{2.5}P{-2.5}}
    \toprule
    \multicolumn{1}{c}{数学常数} &
    \multicolumn{1}{c}{物理常数} \\
    \midrule
    3.14159 & 2.99792 \\
    27.18281 & -17.58819 \\
    \bottomrule
\end{tabular}
\end{table}
```

数学常数	物理常数
3.14159	2.99792
27.18281	−17.58819

例 9.7: 数字表格

9.5 长表格

有时表格太长要跨页，可以使用 Carlisle 的 `longtable` 宏包[5]。这位同志对表格情有独钟，表格的宏包被他承包了一半。我们需要做以下工作：

1. 首先用 `longtable` 环境取代 `tabular` 环境；
2. 然后在表格开始部分定义每页页首出现的通用表头，表头最后一行末尾不用 `\\` 换行，而是加一个 `\endhead` 命令；
3. 接着定义首页表头 (如果它和通用表头不同的话) ，同样地最后一行用 `\endfirsthead` 命令结尾；
4. 然后是以 `\endfoot` 命令结尾的通用表尾；
5. 然后是以 `\endlastfoot` 命令结尾的末页表尾 (如果它和通用表尾不同的话) ；
6. 最后是表格的具体内容。

```
\begin{longtable}{ll}
    \multicolumn{2}{r}{接上页} \\
    \toprule
    作者 & 作品 \\
    \midrule
    \endhead
\caption{长表格} \\
    \toprule
    作者 & 作品 \\
    \midrule
    \endfirsthead
    \bottomrule
    \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。\\
    & 承欢侍宴无闲暇，春从春游夜专夜。\\
    & 后宫佳丽三千人，三千宠爱在一身。\\
    & 金屋妆成娇侍夜，玉楼宴罢醉和春。\\
    & 姊妹弟兄皆列土，可怜光彩生门户。\\
    & 遂令天下父母心，不重生男重生女。\\
    & 骊宫高处入青云，仙乐风飘处处闻。\\
    & 缓歌慢舞凝丝竹，尽日君王看不足。\\
    & 渔阳鞞鼓动地来，惊破霓裳羽衣曲。\\
\end{longtable}
```

表 9.1: 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。

接下页…

接上页

作者	作品
	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。
	承欢侍宴无闲暇，春从春游夜专夜。
	后宫佳丽三千人，三千宠爱在一身。
	金屋妆成娇侍夜，玉楼宴罢醉和春。
	姊妹弟兄皆列土，可怜光彩生门户。
	遂令天下父母心，不重生男重生女。
	骊宫高处入青云，仙乐风飘处处闻。
	缓歌慢舞凝丝竹，尽日君王看不足。
	渔阳鞞鼓动地来，惊破霓裳羽衣曲。

9.6 宽表格

表格太宽时可以使用 Fairbairns [2] 等人的 `rotating` 宏包[6]。其方法很简单，用 `sidewaystable` 环境替代 `table` 环境即可。

```
\begin{sidewaystable}[htbp]
\caption{主流英文词典}
\label{tab:dict}
\centering
\begin{tabularx}{550pt}{Xllcrrr}
    \toprule
    Title & Abbr & Publisher & Year & Pages & Entries &
       Price \\
    \midrule
    Oxford English Dict, 2nd Ed & OED & Oxford Univ
        & 1989 & 21,728 & 616,500 & 995 \\
```

[2] 1970 年代剑桥数学学士，电脑硕士，现任剑桥网管。UK FAQ 的维护者。

```
    \midrule
12  Shorter Oxford English Dict, 7th Ed & SOED & Oxford
        Univ
        & 2007 & 3,888 & 600,000 & 175 \\
14  New Oxford Dict of English, 2nd & NODE & Oxford Univ
        & 2005 & 2,112 & 355,000 & 68 \\
16  Webster's Third New International Dict & W3 & Merriam-
        Webster
        & 1961 & 2,816 & 476,000 & 129 \\
18  American Heritage Dict, 4th Ed & AHD & Houghton
        Mifflin
        & 2000 & 2,112 & 90,000 & 60 \\
20  Random House Webster's Unabridged Dict, 2nd Ed &
        Random & Random House
        & 2005 & 2,256 & 315,000 & 69 \\
22  \midrule
    Concise Oxford Dict, 11th Ed & COD & Oxford Univ
24      & 2006 & 1,728 & 240,000 & \\
    Chambers Dict, 10th Ed & Chambers & Chambers Harrap
26      & 2006 & 1,872 & & 50 \\
    Collins English Dict, 9th Ed & Collins & HarperCollins
28      & 2007 & 1,888 & & 67 \\
    Longman Dict of Contemporary English, 4th Ed & Longman
        & Longman
30      & 2005 & & 207,000 & 71 \\
    Merriam-Webster's Collegiate Dict, 11th Ed & & Merriam
        -Webster
32      & 2003 & 1,664 & 225,000 & 26 \\
    American Heritage College Dict, 4th Ed & & Houghton
        Mifflin
34      & 2007 & 1,664 & & 26 \\
    Random House Webster's College Dict & & Random House
36      & 2005 & 1,632 & & 26 \\
    Webster's New World College Dict, 4th Ed & & John
        Wiley \& Sons
38      & 2004 & 1,744 & 160,000 & 26 \\
    \bottomrule
40 \end{tabularx}
\end{sidewaystable}
```

表 9.2: 主流英语词典

Title	Abbr	Publisher	Year	Pages	Entries	Price
Oxford English Dict, 2nd Ed	OED	Oxford Univ	1989	21,728	616,500	995
Shorter Oxford English Dict, 7th Ed	SOED	Oxford Univ	2007	3,888	600,000	175
New Oxford Dict of English, 2nd	NODE	Oxford Univ	2005	2,112	355,000	68
Webster's Third New International Dict	W3	Merriam-Webster	1961	2,816	476,000	129
American Heritage Dict, 4th Ed	AHD	Houghton Mifflin	2000	2,112	90,000	60
Random House Webster's Unabridged Dict, 2nd Ed	Random	Random House	2005	2,256	315,000	69
Concise Oxford Dict, 11th Ed	COD	Oxford Univ	2006	1,728	240,000	
Chambers Dict, 10th Ed	Chambers	Chambers Harrap	2006	1,872		50
Collins English Dict, 9th Ed	Collins	HarperCollins	2007	1,888		67
Longman Dict of Contemporary English, 4th Ed	Longman	Longman	2005		207,000	71
Merriam-Webster's Collegiate Dict, 11th Ed		Merriam-Webster	2003	1,664	225,000	26
American Heritage College Dict, 4th Ed		Houghton Mifflin	2007	1,664		26
Random House Webster's College Dict		Random House	2005	1,632		26
Webster's New World College Dict, 4th Ed		John Wiley & Sons	2004	1,744	160,000	26

9.7 彩色表格

若想给表格增加点色彩，可以使用 Carlisle 的 `colortbl` 宏包[7]。它提供的 `\columncolor`、`\rowcolor`、`\cellcolor` 命令可以分别设置列、行、单元格的颜色。这三个命令的基本语法相似：

语法：`{颜色}`

`\columncolor` 需要放到列前置命令里，`rowcolor`、`\cellcolor` 分别放到行、单元格之前。`colortbl` 宏包可以使用 `xcolor` 宏包的色彩模型；两者同时，前者不能直接加载，需要通过后者的选项 `table` 来加载。三个命令同时使用时，它们的优先顺序为：单元格、行、列。

```
\usepackage[table]{xcolor}
...
\begin{table}[htbp]
\centering
\begin{tabular}{l>{\columncolor{Yellow}}ll}
    \rowcolor{Red}操作系统  & 发行版    & 编辑器 \\
    Windows                & MikTeX    & TexMakerX \\
    \rowcolor{Green}Unix/Linux & \cellcolor{Lavender}teTeX
        & Kile \\
    Mac OS                 & MacTeX    & TeXShop \\
    \rowcolor{Blue}通用    & TeX Live  & TeXworks \\
\end{tabular}
\end{table}
```

操作系统	发行版	编辑器
Windows	MikTeX	TexMakerX
Unix/Linux	teTeX	Kile
Mac OS	MacTeX	TeXShop
通用	TeX Live	TeXworks

例 9.8: 彩色表格

例 9.8 这样的彩色表格过于花里胡哨，和包老师低调中年宅男的定位极不相称。`xcolor` 宏包的 `rowcolors` 命令 (需要 `colortbl` 宏包的支持) 可以分别设置奇偶行的颜色，甚合吾意。该命令语法如下：

语法：`{起始行}{奇数行颜色}{偶数行颜色}`

例 9.9 中代码第 14 行的 `\hiderowcolors` 命令是用来暂停显示前面设置的奇偶行颜色，否则后面的其他表格会继续显示颜色。另一个命令 `\showrowcolors` 可以用来重新激活奇偶行颜色设置。

```
\usepackage[table]{xcolor}
...
\begin{table}[htbp]
\centering
\rowcolors{1}{White}{Lavender}
\begin{tabular}{lll}
    \hline
    操作系统   & 发行版   & 编辑器 \\
    Windows    & MikTeX   & TexMakerX \\
    Unix/Linux & teTeX    & Kile \\
    Mac OS     & MacTeX   & TeXShop \\
    通用       & TeX Live & TeXworks \\
    \hline
\hiderowcolors
\end{tabular}
\end{table}
```

操作系统	发行版	编辑器
Windows	MikTeX	TexMakerX
Unix/Linux	teTeX	Kile
Mac OS	MacTeX	TeXShop
通用	TeX Live	TeXworks

例 9.9: 彩色表格

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 P. Carlisle. *The tabularx Package*, 1999. URL http://www.ctan.org/tex-archive/macros/latex/required/tools/.

[3] Frank Mittelbach and David P. Carlisle. *A new implementation of LaTeX's tabular and array environment*, 2009. URL http://www.ctan.org/tex-archive/macros/latex/required/tools/.

[4] Wayne A. Rochester. *The warpcol Package*, 2007. URL http://www.ctan.org/tex-archive/macros/latex/contrib/warpcol/.

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

[6] Robin Fairbairns, Sebastian Rahtz, and Leonor Barroca. *A package for rotated objects in LaTeX*, 2010. URL http://www.ctan.org/tex-archive/macros/latex/contrib/rotating/.

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

广告位招租

第十章　结构

> 善张网者引其纲，不一一摄万目而后得。
>
> —《韩非子 · 外储说右下》

在 2.2 节中我们曾简略提及文档的结构，本章将对它们展开介绍。通常一篇文档无论长短都会有标题和正文。如果正文较长，有个层次结构会便于行文，阅读起来也较方便。为了便于查找，长文档通常会前有目录，后有索引。另外为了便于编辑，长文档通常会被分割成多个文件。和 HTML 类似，PDF 也提供超链接功能，它也放在本章介绍。学术文档一般还会有参考文献，以供读者参考、借鉴。

以上诸项通常会按标题、目录、正文、参考文献、索引这样的顺序出现。如果是正式出版的书籍，还会有标题页、版权页、献辞页等；而且这些结构出现的奇偶页也有讲究，比如版权页是偶数页，目录首页是奇数页。同学们若对书籍的格式装帧有兴趣，可以参考 Peter R. Wilson [1] 的 *Notes on Book Design*[1] 和 *Memoir Class*[2]，或者 *Chicago Manual of Style*[3]。

10.1　长文档

文章想要吸引读者就必须写得好，想要影响深远就必须写得长。包子曰：神奇卓异非至人，至人只是长。当文档很长时，我们最好把它分为多个文件，各个击破，从而缓解长期伏案写作带来的悲观厌世情绪。这和高速公路上为了防止疲劳驾驶设置的技术弯道有异曲同工之妙。

[1] 剑桥大学机械学士，诺丁汉大学半导体物理博士。本科毕业后就职于不列颠托马森休斯顿公司 (British Thomson Houston)；后跳槽到卢卡斯研究中心 (Lucas Research Center) 搞 CAD，曾任计算机辅助制造国际 (Computer-Aided Manufacturing International, CAM-I) 欧洲分部主席；再跳槽到 GE参与开发 CAD/CAM 数据交换标准 STEP；又跳槽到 RPI、美国天主教大学 (Catholic University of America)、国家标准技术研究所 (National Institute of Standards and Technology, NIST) 等院所从事科研，最后从波音退休。

例 10.1 示范了如何在主控文档中引用子文档。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}
```

例 10.1: 拆分长文档

当文档很长时，编译一遍也会很花时间。我们可以使用 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
```

10.2 标题

在 例 10.2 中，我们用 \title, \author, \date 等命令分别设置标题、作者、日期。标准文档类没有为作者所属单位定义专门的命令，我们可以把单位直接写在作者下面，作者和单位之间的联系通常用一些特殊符号来标注。设置上述内容之后，我们用 \maketitle 命令来输出它们。

表 2.1 中提到的文档类选项 notitlepage 和 titlepage 可以用来控制标题是否单独占一页。report 和 book 文档类中的标题缺省独占一页，article 文档类中的标题缺省和正文等混居一页。

10.3 目录

\tableofcontents 命令可以用来生成整个文档的章节目录。LaTeX 会自动设定目录包含的章节层次，我们也可以用 \setcounter 命令来指定目录的层次深度。如果不想让某个章节标题出现在目录里，则可以使用 例 10.3 中带 * 的命令来声明章节。

类似地，我们可以使用 \listoffigures 和 \listoftables 命令来生成图目录和表目录。

```
\title{雷人的传说}
\author{包太雷$^*$\quad 包巨雷$^\dagger$\quad
    包最雷$^\ddagger$\\[10pt]
$*$Barrington University, Burlington, VT\\
$\dagger$Pacific Western University, San Diego, CA\\
$\ddagger$Preston University, Los Angeles, CA}
\date{2011年1月11日}
\maketitle
```

例 10.2: 标题

```
\tableofcontents
\setcounter{tocdepth}{2}
\chapter*{...}
\section*{...}
\subsection*{...}
```

例 10.3: 目录

当章节或图表等结构发生变化时，我们需要执行两次编译命令以获得正确的目录。其中第一次编译生成一些中间文件，后缀分别为 `.toc` (目录)、`.lof` (图目录)、`.lot` (表目录)；第二次编译则把这些中间文件和其他内容整合起来。LaTeX 之所以设计成这样可能是因为当时的电脑内存容量有限。

另外我们也可以利用 Axel Sommerfeldt 的 `caption` 宏包[4] 来自定义类似于插图和表格的浮动环境。本文中的例子浮动环境就是用 例 10.4 中的方法生成的，第二行代码指定 `loe`为例目录中间文件后缀，`example` 为环境名，例为浮动环境标题前缀，例目录是目录的标题。定义该环境后，其使用方法与插图、表格浮动环境一样，见第 3–5 行。

语法：\DeclareCaptionType[选项]{环境}[名称][目录名]

```
\DeclareCaptionType[fileext=loe]{example}[例][例目录]
\begin{example}[h]
...
\end{example}
```

例 10.4: 自定义浮动环境

10.4 参考文献

10.4.1 thebibliography

在学术文档中人们经常要用到参考文献 (bibliography)，这样做既可以有选择地提供事实，作客观公证科学严谨状，还可以拉帮结派党同伐异。

LaTeX 中最原始的方法是用 thebibliography 环境和 \bibtem 命令来定义参考文献条目。在 例 10.5 中，第一行的参数 9 是参考文献条目编号的宽度；如果有几十个条目，可以把该参数改为 99。

```
\begin{thebibliography}{9}
\bibitem{Rowling_1997}
    Joanne K. Rowling,
    \emph{Harry Potter and the Philosopher's Stone}.
    Bloomsbury, London,
    1997.
\end{thebibliography}
```

参考文献

[1] Joanne K. Rowling, *Harry Potter and the Philosopher's Stone*. Bloomsbury, London, 1997.

例 10.5: thebibliography 环境

thebibliography 环境一般放在文档的末尾。定义了参考文献之后，我们可以用 \cite 命令在正文中引用条目。

```
\cite{Rowling_1997}
```

[1]

10.4.2 BibTeX

`thebibliography` 环境的一个缺点是，用户得自己调整显示格式，这样做很麻烦而且易出错。

Oren Patashnik (1954–) [2] 和 Lamport 就在 1985 年想出一个办法，用数据库文件 `.bib` 记录参考文献条目，用样式文件 `.bst` 设置显示格式。普通用户一般不需要改动样式文件，只须维护数据库。

这种方法秉承了 LaTeX 内容与格式分离的思想，我们在 SGML/DSSSL, HTML/CSS, XML/XSL 等技术上也可以见到同样的思路。

BibTeX 将参考文献分为十几种类型，每种类型的参考文献有不同的必选项和可选项 (见以下列表) 。

article 期刊或杂志上的文章
- 必选项：author, title, journal, year
- 可选项：volume, number, pages, month, note

conference 同 inproceedings

book 正式出版的书籍
- 必选项：author/editor, title, publisher, year
- 可选项：volume/number, series, address, edition, month, note

booklet 非正式出版的小册子
- 必选项：title
- 可选项：author, howpublished, address, month, year, note

inbook 书的一部分，比如章、节，或某些页
- 必选项：author/editor, title, chapter/pages, publisher, year
- 可选项：volume/number, series, type, address, edition, month, note

incollection 书中比较独立的一部分
- 必选项：author, title, booktitle, publisher, year
- 可选项：editor, volume/number, series, type, chapter, pages, address, edition, month, note

inproceedings 会议论文
- 必选项：author, title, booktitle, year
- 可选项：editor, volume/number, series, pages, address, month, organization, publisher, note.

manual 手册
- 必选项：title

[2] 1976 年毕业于耶鲁，1990 年斯坦福 Knuth 门下电脑博士。1980 年加入贝尔实验室。

- 可选项：author, organization, address, edition, month, year, note

mastersthesis 硕士论文

- 必选项：author, title, school, year
- 可选项：type, address, month, note

misc 实在不好分类时只好用它

- 必选项：无
- 可选项：author, title, howpublished, month, year, note

phdthesis 博士论文

- 必选项：author, title, school, year
- 可选项：type, address, month, note

proceedings 会议论文集

- 必选项：title, year
- 可选项：editor, volume/number, series, address, month, organization, publisher, note

techreport 技术报告

- 必选项：author, title, institution, year
- 可选项：type, number, address, month, note

unpublished 未出版文档

- 必选项：author, title, note
- 可选项：month, year

编辑 `.bib` 文件时可以用普通文本编辑器，也可以用专门的文献管理软件来提高效率，后者包老师推荐 JabRef。一些其他的文献管理软件或网络服务也可以输出 `.bib` 格式，比如 EndNote, Google Scholar, Zotero 等。

例 10.5 中罗琳阿姨的书可以用 BibTeX 改写成 例 10.6 中的样子。其中每行是一个数据项，第一个数据项是关键字，供引用时用；其他数据项都以名称=值 的形式成对出现，值要写在双引号之内；数据项之间用逗号分隔。

```
@book{Rowling_1997,
    author    = "Joanne K. Rowling",
    title     = "Harry Potter and the Sorcerer's Stone",
    publisher = "Bloomsbury, London",
    year      = "1997"
}
```

例 10.6: BibTeX 数据

有了数据后，我们需要选一个样式。通常的 LaTeX 发行版都会带有四种标准的样式，如果觉得这些标准格式无法满足你的需要，可以参考 Nicolas

Markey (1976–) [3] 的《野兽调教》[5]。

plain 参考文献列表按作者姓氏排序，序号为阿拉伯数字。
unsrt 参考文献列表按正文中引用顺序排序，序号为阿拉伯数字。
alpha 参考文献列表按作者姓氏排序，序号为作者姓氏加年份。
abbrv 类似 `plain` 样式，作者名字、月份、期刊名等用缩写。

选定样式后，我们需要在文档中用 `\bibliographystyle` 命令来设置样式，然后用 `\bibliography` 命令输出参考文献列表。

```
\bibliographystyle{plain}
\bibliography{myref}
```

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，它需要依次执行等四次编译操作。

1. 第一遍 `xelatex` 把参考文献条目的关键字写到中间文件 `.aux` 里去。
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 `\bibtem` 命令。
3. 第二遍 `xelatex` 把交叉引用写到 `.aux` 中去。
4. 第三遍 `xelatex` 则在正文中正确地显示引用。

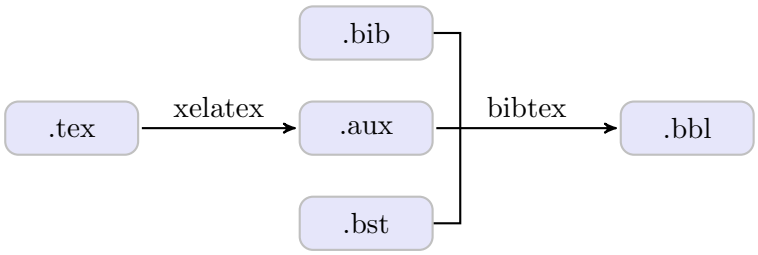


图 10.1: BibTeX 的编译

有多个子文档时，我们可以在每个子文档中用 `\bibliographystyle` 命令设置不同的样式；当然如果没有特别的理由，包老师还是建议用统一的样式。编译时用 `xelatex` 编译主控文档，而用 `bibtex` 编译各个子文档。

[3] 1994 年数学学士，1998 年巴黎第七大学 (Paris Diderot University) 数学和电脑双硕士，2003 年奥尔良大学 (University of Orléans) 电脑博士。布鲁塞尔自由大学博士后，2004 年加入 CNSR。

```
xelatex master(.tex)
bibtex chapter1(.tex)
bibtex chapter2(.tex)
...
xelatex master(.tex)
xelatex master(.tex)
```

例 10.7: 子文档参考文献的编译

10.4.3 Natbib

参考文献在正文中的引用通常有两种模式：作者 -年份和数字。LaTeX 提供的 `\cite` 命令只支持数字模式，Patrick W. Daly [4] 的 `natbib` 宏包[6] 则同时支持这两种模式。

`natbib` 提供了三种列表样式：`plainnat`，`abbrvnat`，`unsrtnat`，它们的参考文献列表和相对应的 LaTeX 标准样式 `plain`，`abbrv`，`unsrt` 效果相同，只是在引用时可以自由选择作者 -年份或数字模式。

这两种模式以及他一些细节的设置 (比如标点符号) 在本文中被称作引用样式。`natbib` 的三种列表样式都有自己的缺省引用样式，如要定制引用样式，可以使用 `\setcitestyle` 命令；其选项见 表 10.1，其中上标模式其实就是把数字标号移到了上标位置。。

表 10.1: 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

`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 LaTeX 本身提供的 `\cite` 命令，因为它在作者 -年份模式下和 `\citet` 效果相同，在数字模式下和 `\citep` 相同。这些模式下引用命令的效果见 例 10.8，

另外还有一些引用命令，如 `\citetext`、`\citenum`、`\citeauthor`、`\citeyear` 等，读者可以自行查阅手册，此处不赘述。

[4] 马克斯 · 普朗克研究所研究员。

```
\setcitestyle{authoryear}
see \cite{Daly_2010}\\
see \citet{Daly_2010}\\
see \citep{Daly_2010}
```

see Daly [2010]
see Daly [2010]
see [Daly, 2010]

```
\setcitestyle{numbers}
see \cite{Daly_2010}\\
see \citet{Daly_2010}\\
see \citep{Daly_2010}
```

see [6]
see Daly [6]
see [6]

```
\setcitestyle{super}
see \cite{Daly_2010}\\
see \citet{Daly_2010}\\
see \citep{Daly_2010}
```

see[6]
see Daly[6]
see[6]

例 10.8: 各种引用模式下的引用命令效果

10.5 索引

makeidx 宏包提供了索引功能。应用它时，我们首先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}
```

例 10.9: 索引

当编译含索引的文档时，用户需要执行三次编译操作，

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

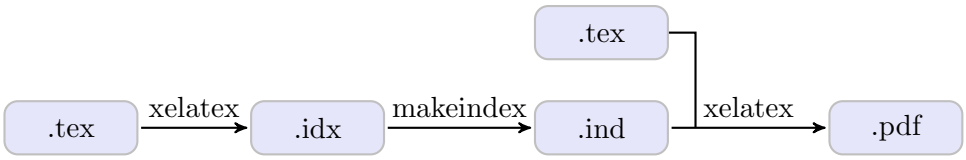


图 10.2: 索引的编译

10.6 超链接

Sebastian P. Rahtz [5] 和 Heiko Oberdiek [6] 的 `hyperref` 宏包[7] 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，还提供了其他几个命令。

`\hyperref` 命令对已经定义的标签进行简单包装，加上文字描述。

```
\label{sec:hyperlink}
...
编号形式的链接：\ref{sec:hyperlink}\\
文字形式的链接：\hyperref[sec:hyperlink]{链接}
```

编号形式的链接：10.6
文字形式的链接：链接

例 10.10: `\hyperref` 命令

`\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

例 10.11: `\url` 和 `\href` 命令

[5] 1970 年代牛津大学希腊语学士，考古学硕士。1980 年代南安普敦大学人类学讲师，后跳槽到 CERN、爱思维尔出版公司 (Elsevier)，现任牛津大学信息主管。TUG 和 CTAN 的重要成员。

[6] pdfTeX 开发者之一，几十个宏包的作者。

10.7 结构名

在 LaTeX 中，每个文档结构都有自己的名字，一般用来在标题中或引用时显示。比如主目录、图目录、表目录的名字分别是：Contents, List of Figures, List of Tables；章、节、小节的名字分别是：Chapter, Section, Subsection；图和表的名字是 Figure 和 Table。

例 10.12 列出了标准文档类中定义的结构名变量，其中 `\bibname` 是 `book` 文档类的专有变量；`\abstractname` 和 `\refname` 是 `report` 和 `article` 文档类专有的；其他变量对这三种文档类都适用。

如果我们想改变这些变量的值，比如中文文档需要中文结构名，可以用 例 10.12 中的方法来重定义这些结构名变量。例中第四、五行代码是为了顺应中文的习惯。

```
\renewcommand{\contentsname}{目录}
\renewcommand{\listfigurename}{图目录}
\renewcommand{\listtablename}{表目录}
\renewcommand{\partname}{第 \thepart 部}
\renewcommand{\chaptername}{第 \thechapter 章}
\renewcommand{\figurename}{图}
\renewcommand{\tablename}{表}
\renewcommand{\bibname}{参考文献}
\renewcommand{\appendixname}{附录}
\renewcommand{\indexname}{索引}
\renewcommand{\abstractname}{摘要}
\renewcommand{\refname}{参考文献}
```

例 10.12: 标准文档类结构名重定义

2.9 节中提到的 `\ref` 命令显示的是数字。`hyperref` 宏包提供了一个 `\autoref` 命令，它可以自动判断标签所属结构对象的类型，为引用加上合适的名字，输出时显示结构名加上结构编号。该宏包也为此定义了一些结构变量名，我们也可以用同样的方法重定义它们 (见 例 10.13) 。

需要注意的是，`\autoref` 命令输出的结果总是名称在编号前面，对于章、节等结构无法产生“第 x 章”、“第 x 节”等符合中文习惯的结果。所以 例 10.13 略去了若干这样的结构名，我们在引用时需要手工在 `\ref` 命令前后加上合适的字眼。

```
\renewcommand{\equationautorefname}{公式}
\renewcommand{ \footnoteautorefname}{脚注}
\renewcommand{\itemautorefname}{项}
\renewcommand{\figureautorefname}{图}
\renewcommand{\tableautorefname}{表}
\renewcommand{\appendixautorefname}{附录}
\renewcommand{\theoremautorefname}{定理}
```

例 10.13: `hyperref` 宏包结构名重定义

参考文献

[1] Peter R. Wilson. *A Few Notes on Book Design*, 2009. URL http://www.ctan.org/tex-archive/info/memdesign/.

[2] Peter R. Wilson. *The Memoir Class*, 8th edition, 2010. URL http://www.ctan.org/tex-archive/macros/latex/contrib/memoir/.

[3] UCP, editor. *Chicago Manual of Style, 15th Edition*. University of Chicago, 2003. URL http://en.wikipedia.org/wiki/Chicago_Manual_of_Style.

[4] Axel Sommerfeldt. *Customizing captions of floating environments using the caption package*, 2008. URL http://ctan.org/tex-archive/macros/latex/contrib/caption/.

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

[6] Patrick W. Daly. *Natural Sciences Citations and References*, 2010. URL http://www.ctan.org/tex-archive/macros/latex/contrib/natbib/.

[7] Sebastian Rahtz and Heiko Oberdiek. *Hypertext Marks in LaTeX: A Manual for hyperref*, 2010. URL http://www.tug.org/applications/hyperref/.

第十一章　布局

> 离娄之明，公输子之巧，不以规矩，不能成方圆。
>
> —《孟子 · 离娄上》

我们在 2.8 节中曾经提到，在 LaTeX 中每一个排版对象都是一个盒子。排版就是要把小盒子用空白间距粘在一起放到大盒子里，然后再依次嵌套到更大的盒子里。怎样优化这些大大小小的盒子是一门很深的学问，在本章里我们将会略窥门径。

11.1　页面尺寸

在排版时页面是最大的盒子，所以我们就先看一看它。人们在日常生活中可以见到多种规格的纸张，它们一般归属于两大类标准：公制和美制。

11.1.1　普通青年

早在 1786 年，德国科学家 Georg C. Lichtenberg (1742–1799)就发现 $1/\sqrt{2}$ 这个比例可以用于分割纸张。你拿一张这个比例的纸，在较长的那个方向上一分为二，得到的两张纸也是同样比例。

1912–1914 年间，另一个德国人 Walter Porstmann (1886–1959)在为诺贝尔化学奖得主 Wilhelm Ostwald (1853–1932) [1] 当助手时，两人企图把这个比例变成一个世界标准。一个数学家和一个化学家在一起琢磨这个好像不务正业，但是考虑到 Ostwald 当时还兼任一个知识产权研究所的主任，好歹也沾点边儿。他们从 1cm x 1.41cm 开始，每次加倍短边，直至无穷。然而他们的提议无人理会，也许人们都正忙于第一次世界大战。此后几年 Porstmann 一直沿着这个方向灌水，甚至他在 1918 年写的博士论文也与此相关。

[1] 生于爱沙尼亚，1878 年塔图大学 (University of Tartu) 博士，1909 年诺贝尔化学奖。

当时德国标准化学会 (Deutschen Instituts für Normung, DIN) 刚成立没多久，无所事事的学会领导 Waldemar Hellmich 注意到了 Porstmann，就在 1920 年将其网罗至门下。1922 年，DIN 发布了 476 纸张标准，这次是从面积一平方米的 A0 (841mm x 1189mm) 开始，每次减半长边。后来又陆续加上了 B, C, D 系列。

DIN 476 因为其简单易用，逐渐流行到其他国家。1961 年 ISO 将 A 和 B 系列采纳为推荐标准，1975 年变成 ISO 216 标准。其中 B 系列比 DIN 476 的略大一点，它从 1000mm x 1414mm 的 B0 开始。1985 年发布的 ISO 269 加上了 C 系列，它的尺寸是 A 和 B 系列纸张尺寸的几何平均。

A 系列常用于公文；B 系列常用于海报和护照 (B7, 88mm x 125mm)；C 系列常用于信封，因为它恰好比 A 系列大一点，比如 A4 纸可以装在 C4 信封里，对折一下就可以装进 C5 信封，再对折一次装进 C6 信封。

11.1.2 文二青年

当今世界大多数国家都采用此项国际标准，冥顽不化的只有美国和它的几个小弟，在这些地方流行的是比 A4 胖一点的 Letter (8.5in x 11in)。Letter 追溯上去应该源于大英帝国，只是英国在 1950 年代末就已经全面接受了公制标准。窃以为一小撮美洲人对英制情有独钟的原因是他们的历史短，所以但凡有把年纪的东西都颇为珍惜。

IEEE定义过一个 Government-Letter (8in x 10.5in)，主要用于儿童读物。想来儿童读的东西信息量少，不需要 Letter 那么大的纸。Herbert Hoover (1874–1964) [2] 在 1920 年代担任商务部长期间，规定政府公文也采用此规格。我今天量过《时代周刊》和《商业周刊》，也是这个规格。

1980 年代初 Ronald Reagan (1911–2004) 上台后，认为美利坚乃天朝上国，用小纸太丢脸，于是政府公文改用 Letter。我对 Letter 没啥意见，在写本章之前用的就是它，现在改用 A4 是为了照顾中文读者的习惯。坑爹的是 Legal (8.5in x 14in)，比普通文件夹都长一大截，它主要用于法律文件。

面对世界标准化潮流，老美的颜面有点 hold 不住，于是在 1996 年推出 ANSI Y14.1 作为遮羞布。它定义了 A, B, C, D, E 五种规格，A 就是 Letter，B 比 A 面积大一倍，C 比 B 大一倍，依次类推。它们的长宽比不一致，B 和 C 比其他三种瘦很多。它们的尺寸倒是和 A4–A0 差不多，如果不挑剔也可以混用。

[2] 美国第 31 任总统，历任总统中罕见的工科 WSN。1895 年斯坦福地质学士，自称是该校首名学生。在校期间曾担任棒球队经理，向前总统 Benjamin Harrison追讨两毛五门票钱。

11.1.3 尺寸详解

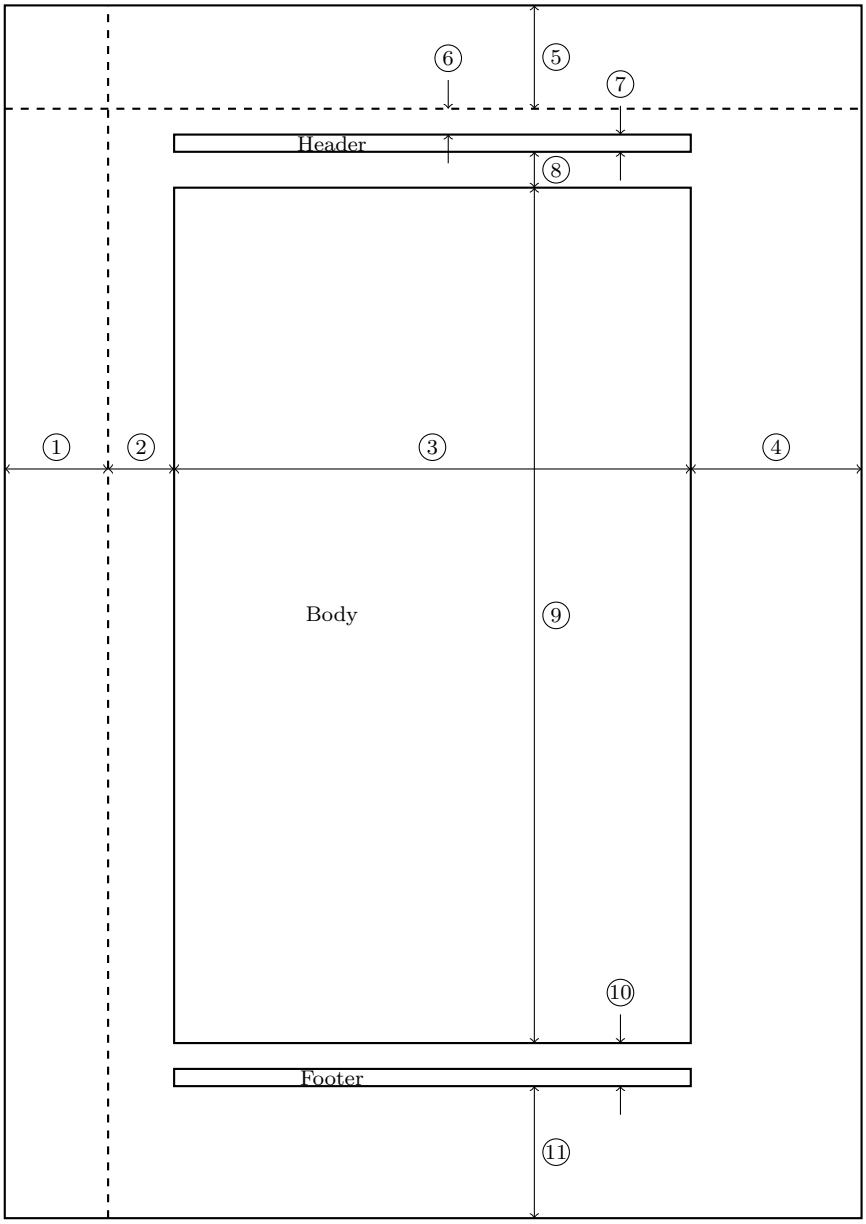


图 11.1: 页面尺寸

图 11.1 是一张 A4，尺寸为 210mm x 297mm 也就是 597pt x 845pt。图中标注为 Body 的是正文区域，Header 是页眉，Footer 是页脚。图中标注的尺寸有些是固定的，另一些是随着正文字号或其他因素而改变的；在这里我们会遇到一些 LaTeX 定义的尺寸宏变量。

lshort 里有张类似的图流传甚广，包老师为了操练一把 20 年前学到的机械制图本领，就自己画了一张。图中尺寸用的是 11pt,oneside 的，下面我们就以此为例介绍一下这些尺寸：

1. 页边距，1in。在微软 Word 里这个尺寸也很常见；
2. `\oddsidemargin` 或 `\evensidemargin`，奇数或偶数页左边距，46pt；
3. `\textwidth`，正文宽度，360pt，可以放下大概 32 个汉字；
4. 597pt 减去左边的 1in + 46pt 和中间的 360pt，还剩下 119pt，左右相差不到 1pt。如果双面打印的话，两面的正文部分恰好是重叠的；
5. 页边距，1in；
6. `\topmargin`，上边距，18pt；
7. `\headheight`，页眉高度，12pt；
8. `\headsep`，页眉与正文间距，25pt；
9. `\textheight`，正文高度，595pt，可以放下 38 行文字；
10. `\footskip`，正文与页脚基线间距，30pt。它比页眉的 12pt + 25pt 小了 7pt，不理解的同学可以照照镜子，你左右是对称的，但是上下呢？
11. 845pt 减去上面全部尺寸，还剩下 93pt，比上面的 1in + 18pt 多了 3pt。

当字号等发生变化时，上述某些尺寸也会发生一定的变化。比如我们把 oneside 改成 twoside，那么奇偶页的左边距就分别变成 22pt 和 70pt。但是奇数页右边空白恰好和偶数页左边空白相等，不会给双面打印造成困扰。

一般情况下我们无须改动 LaTeX 的页面布局缺省设置。当有特殊需要时，可以使用 section 2.4 节提到的 `\setlength` 或 `\addtolength` 来设置上述宏变量的值。

梅木秀雄[3]的 `geometry` 宏包[1] 则提供了更高级的用户接口。比如我们可以用如下的方法来设置页面尺寸和边距，

```
\usepackage[paperwidth=100mm, paperheight=150mm, margin=20
    mm]{geometry}
```

也可以像下面这样单独设置每个边距，

[3]东芝高级经理。

```
\usepackage[top=2in, bottom=1in, left=1in, right=1in]{
    geometry}
```

想把页面横过来，可以这样，

```
\usepackage[landscape]{geometry}
```

11.2 页面样式

了解页面尺寸之后，我们再来看看页面样式，也就是页眉和页脚的内容。LaTeX 预定义了四种样式，见 表 11.1。

表 11.1: LaTeX 页面样式

empty	页眉、页脚空白
plain	页眉空白，页脚含居中页码，非 book 文档类缺省值
headings	页脚空白，页眉含章节名和页码，book 文档类缺省值
myheadings	页脚空白，页眉含页码和用户自定义信息

我们可以用 \pagestyle 和 \thispagestyle 命令来设置整个文档或单独某页的样式。

```
\pagestyle{plain}     %全局设置
\thispagestyle{empty}%单页设置
```

表 11.1 中的样式是通过重定义四个宏变量 @oddhead, @evenhead, @oddfoot, @evenfoot 来设置奇偶页的页眉与页脚。

我们也可以用 例 11.1 中的方法来自定义样式。其中第二行代码定义了 permanentdamagedhead 样式，定义这个特殊命令时一定要写成 \ps@style 的样子；而在引用时，则写成 \pagestyle{style}。第三至六行分别定义了奇偶页的页眉和页脚；单面文档奇偶页样式一样，所以需要且只需要定义奇数页的页眉和页脚，偶数页的定义不起作用。

这段代码中的 \hfill 是个弹性填充命令，它把两边的内容“推”得尽可能远。例中使用了特殊符号 @，所以要在第一行用 \makeatletter 命令声明一下，暂时把它当正常符号用；用完之后，在最后一行用相应的 \makeatother 命令恢复现场。

在自定义页面样式时，我们不仅可以在页眉和页脚里使用普通字符串，也可以使用一些宏变量来显示页码、章节号码和名称等（见 表 11.2）。

```
\makeatletter
\newcommand{\ps@permanentdamagedhead}{
    \renewcommand{\@oddhead}{信春哥\hfill 不挂科}
    \renewcommand{\@oddfoot}{\hfill\thepage\hfill}
    \renewcommand{\@evenhead}{芙蓉姐姐\hfill 美若天仙}
    \renewcommand{\@evenfoot}{\@oddfoot}
}
\makeatother
```

信春哥 不挂科

至于你信不信，我反正信了。

1

芙蓉姐姐 貌似天仙

闭月羞花，沉鱼落雁。

2

例 11.1: 自定义页面样式

表 11.2: 页眉和页脚常用宏变量

`\thepage`	页码
`\thechapter`	章编号
`\thesection`	节编号
`\chaptername`	章起始单词名，Chapter
`\sectionname`	节起始单词名，Section
`\leftmark`	左标记，在 `article` 文档类中包含 `section` 信息，在 `report` 和 `book` 中则包含 `chapter` 信息。
`\rightmark`	右标记，在 `article` 中包含 `subsection` 信息，在 `report` 和 `book` 中则包含 `section` 信息。

表 11.2 中前五个变量都可以直接重定义，左右标记特殊一点，需要用下面的命令来间接定义，

```
\markboth{左标记}{右标记}%定义两个标记
\markright{右标记}       %定义右标记
```

需要注意的是，引用时要用`\leftmark` 和 `\rightmark`，定义时要用 `\markboth` 和 `\markright`。为什么会这样呢？因为 Lamport 是个怪叔叔。另外为什么没有 `\markleft` 呢？其实这个命令在 AMS 的几个文档类里是有

的。这个故事告诉我们：AMS 是左派进步青年，Lamport 是右倾分子，所以他后来脱离革命加入了微软。

例 11.2 是 book 文档类中双面打印时 headings 样式定义的节选。其中第二行清空页脚；第三行定义偶数页（也就是左页）页眉，页码居左，左标记居右；第四行定义奇数页（右页）页眉，右标记居左，页码居右。这样页码总是在书的外侧，便于阅读时翻页定位。

```
\def\ps@headings{%
    \let\@oddfoot\@empty\let\@evenfoot\@empty
    \def\@evenhead{\thepage\hfil\slshape\leftmark}%
    \def\@oddhead{{\slshape\rightmark}\hfil\thepage}%
    ...
```

例 11.2: headings 样式

例 11.3 则是 book 文档类中 myheadings 样式的定义。只看前四行它好像和 headings 样式是孪生兄弟，差别在于下面三行清空了左右标记，用户需要自行定义，否则偶数页右上角和奇数页左上角是空白的。

```
\def\ps@myheadings{%
    \let\@oddfoot\@empty\let\@evenfoot\@empty
    \def\@evenhead{\thepage\hfil\slshape\leftmark}%
    \def\@oddhead{{\slshape\rightmark}\hfil\thepage}%
    \let\@mkboth\@gobbletwo
    \let\chaptermark\@gobble
    \let\sectionmark\@gobble
}
```

例 11.3: myheadings 样式

例 11.4 演示了怎样为 myheadings 样式定制左右标记。写到这里包老师忽然想到，好像东西方文化在左比右高级这一点上是一致的，文东武西，男左女右。正因为如此阳顶天挂了之后杨逍可以去泡妞，而范遥只能毁了容去卧底，思想境界上还是差那么一点点。

用户可能会觉得上述定义页面样式的方法比较繁琐和低级，Piet van Oostrum[4]的 fancyhdr 宏包[2] 则提供了更灵活的控制和高级的语法。

在 例 11.5 的代码中，第三行将页面样式设置为 fancyhdr 宏包定义的 fancy 样式；第五至十行分别定制了页眉和页脚；最后两行定制了页眉下方和页脚上方的横线。

[4]曾任教于荷兰乌德勒支大学（Utrecht University），退休后隐居于南美。

```
\documentclass{book}
\markboth{光明左使}{光明右使}
\pagestyle{myheadings}
...
\begin{document}
范遥
\newpage
杨逍
\newpage
\end{document}
```

光明右使 1

范遥

2 光明左使

杨逍

例 11.4: 定制左右标记

```
\usepackage{fancyhdr}
...
\pagestyle{fancy}
\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

例 11.5: `fancyhdr` 宏包

左右标记里除了页码，通常还会有章节编号和名称等，这时我们就要用到另两个命令 `\chaptermark` 和 `\sectionmark`，其用法见 例 11.6。

在这段代码中，第三行清空了页眉和页脚；第四行将页码置于偶数页

左上和奇数页右上，第五行将左标记置于偶数页右上，第六行将右标记置于奇数页左上。

```
\usepackage{fancyhdr}
\pagestyle{fancy}
\fancyhf{}
\fancyhead[LE,RO]{\thepage}
\fancyhead[RE]{\leftmark}
\fancyhead[LO]{\rightmark}
\fancypagestyle{plain}{
    \fancyhf{}
    \renewcommand{\headrulewidth}{0pt}
}
\renewcommand\chaptermark[1]{\markboth{\chaptername\ \
   thechapter: #1}{}}
\renewcommand\sectionmark[1]{\markright{\thesection: #1}}
```

例 11.6: 定制章节标记

`fancyhdr` 宏包将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。代码第七至十行清空了 `plain` 页的页眉和页脚，还去掉了页眉下方的横线。

第 11 行重定义了章标记，它传给 `\markboth` 命令的参数是章的名称，这样章名就会出现在左标记里。章名最初是从哪里来的呢？它是用 `\chapter{章名}` 命令定义的。章名从 `\chapter{}` 传到 `\chaptermark`，再传到 `\markboth`，最后到了 `\leftmark` 那里。它这样一路走来，不畏艰险，踏平坎坷，战胜妖魔，最终修成正果。

类似地，第 11 行重定义了节标记，它传给 `\markright` 命令的参数是节的名称，这样节名就会出现在右标记里。

11.3 分栏

我们经常看到一些期刊报纸使用两栏或更多的栏位，这样可以节省空间和方便阅读。LaTeX 的标准文档类都有一个选项来支持双栏，

```
\documentclass[twocolumn]{article}
```

Mittelbach 的 multicol 宏包提供了更多的功能，比如它支持多达十个栏位，栏位数目可以任意切换；各栏长度相同，最后一页看起来会左右平衡一些。

例 11.7 中第二行代码将栏位之间的距离设为 12pt（缺省是 10pt），第三行将栏位之间的分割线宽度设为 1pt（缺省 0pt，也就是不显示）；下面几行使用了 multicols 环境，注意环境名和宏包名是不同的。

```
\usepackage{multicol}
\setlength{\columnsep}{12pt}
\setlength{\columnseprule}{1pt}
\begin{multicols}{2}
...
\end{multicols}
```

如是我闻，一时，佛在舍卫国祇树给孤独园，与大比丘众千二百五十人俱。尔时，世尊食时，著衣持钵，入舍卫大城乞食。于其城中，次第乞已，还至本处。饭食讫，收衣钵，洗足已，敷座而坐。

例 11.7: multicol 宏包

multicols 环境对浮动体的支持有限，只能使用带 * 的版本，见例 11.8。这种特殊浮动体是跨栏位的，而且它们的 h 选项会失效，也就是不会出现在当前页。不管你的期望是多么地殷切，它最早也只能出现在下一页的页首。

```
\begin{figure*}[tbp]
...
\end{figure*}

\begin{table*}[tbp]
...
\end{table*}
```

例 11.8: 特殊浮动体

某些权宜之计可以让浮动体出现在某栏内，但是它们就失去了浮动性，所以此处不赘述。

11.4 分页

TEX 通常都会自动分页，无须人工干涉。但是浮动体较多的情况下，分页就变成一个 NP 完全问题 [5]，自动分页的效果可能不是我们想要的。这时就需要手工插入分页命令，

```
\newpage
```

如果我们也不确定某处分页是否妥当，可以使用另一个命令，给 TEX 留点面子。这个参数取值 1–4，4 表示强烈要求分页，1 表示你看着办吧。

```
\pagebreak[3]
```

类似地，我们还可以建议 TEX 不要分页，其参数取值也是 1–4，4 表示强烈反对分页，1 表示随便。

```
\nopagebreak[2]
```

浮动体较多，TEX 无所适从时，我们可以用下面的命令帮它减轻点责任。此命令要求 TEX 排完此前所有浮动体，不管是否难看，咱就这么办了。

```
\clearpage
```

参考文献

[1] Hideo Umeki. *The geometry package*, 2010. URL http://ctan.org/tex-archive/macros/latex/contrib/geometry/.

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

[5] 传说 Knuth 在 2046 年写完那套书后，才会有时间改进分页算法。

广告位招租

第十二章　应用

12.1　幻灯

12.2　海报

12.3　书信

12.4　简历

12.5　棋谱

广告位招租

附录 A　软件安装

表 A.1: 软件链接

MacTeX	http://www.tug.org/mactex/
MikTeX	http://www.miktex.org/
TeX Live	http://www.tug.org/texlive/
Kile	http://kile.sourceforge.net/
TeXShop	http://www.uoregon.edu/~koch/texshop/
TeXstudio	http://texmakerx.sourceforge.net/
TeXworks	http://www.tug.org/texworks/
Emacs	http://www.gnu.org/software/emacs/
PSPad	http://www.pspad.com/
Vim	http://www.vim.org/
Ghostscript	http://www.ghostscript.com/
GSview	http://pages.cs.wisc.edu/~ghost/
Bullzip PDF Printer	http://www.bullzip.com/products/pdf/info.php
ImageMagick	http://www.imagemagick.org/
Paint.NET	http://www.getpaint.net/
JabRef	http://jabref.sourceforge.net/

广告位招租

附录 B　印刷简史

> Printing is a human achievement that has demonstrated far greater power to shape the world than all the forces of mordern weaponry.
>
> — John F. Kennedy

人类和其他动物的差别之一是语言和文字，而语言文字使得知识的记载、积累和传承成为可能。印刷的出现促进了大规模知识传播，是人类文明史上的里程碑。

一般而言，印刷 (printing) 包含四种要素：印版 (image carrier) 、印刷材料、墨、印刷设备；以及两大过程：排版 (typesetting) 和压印 (presswork) 。排版负责转换原始的图像和文字，在印版上制作印纹；而压印则负责给印纹上墨，在印刷材料上印制出图文。有时人们会把压印和印刷这两个词混用，我们姑且认为广义的印刷包括排版和狭义的印刷。

当然以上只是一种笼统概括的说法。比如打印机并没有印版，至少没有物理印版。但是我们可以认为图像输出到打印机之前，在电脑内是以虚拟印版的形式存在的。也就是说打印机达到了“机中无版，心中有版”的境界。

窃以为印版是印刷的核心，而数字电脑的应用是印刷工业的一个分水岭，所以下面分四个部分 (传统印刷、传统排版、数字印刷、数字排版) ，按印刷排版技术发展的大致顺序对它们作简单介绍。这里主要谈印版和排版，略谈打印机，基本不涉及印刷材料、墨、印刷机等，因为这些议题和 LaTeX 关系太远。

B.1　传统印刷

传统印刷按过程和和方法可以分为四大类：

1. 凸版印刷 (relief printing)，印纹凸出印版，凸纹表面上墨，压印到印刷材料上。包括雕版 (block)、活字 (moveble type)、活版 (letterpress)、柔版 (flexography) 等技术。

2. 凹版印刷 (intaglio printing)，印纹部分凹下，凹纹中灌墨。包括雕刻 (engraving)、针刻 (drypoint)、磨刻 (mezzotint)、蚀刻 (etching)、细点蚀刻 (aquatint) 等方法。

3. 平版印刷 (planography)，印纹和非印纹部分在同一个平面上，利用油水相斥原理或感光材料上墨。包括石版 (lithography)、石版胶印 (offset lithography) 等技术。

4. 孔版印刷 (porous printing)，印版上有孔，油墨从其中渗漏到印刷材料上。包括镂空 (stenciling)、丝网印刷 (screen-printing) 等方法。

B.1.1 凸版印刷

雕版可以追溯到东汉末年，也就是公元 220 年以前。当时工匠们把文字和图形雕刻在整块木版上，然后涂上墨或染料，再印到丝织品或布料上。木版的缺点是易磨损而难修改。隋唐年间，雕版印刷一度盛行。现存最早的雕版印刷品是唐代 868 年的《金刚经》(图 B.1)。[1]

北宋宝元年间，湖北印刷工毕升 (970–1051) 在 1040 年左右发明了活字印刷术。他用胶泥刻字，火烧成陶；然后将活字排入铁版，整版印刷。其后在中国和朝鲜都曾出现过金属活字。

活字印刷术后来经由俄罗斯和阿拉伯传入欧洲。在东方技术的影响下，德国出版家 Johannes Gutenberg (1398–1468) 于 1450 年前后发明了活版印刷术，包括铅字、油墨和印刷机 (printing press) 等技术。

Gutenberg 同学用油墨取代了水墨，使得印刷品更耐久。传说他年轻时做过金匠，拥有丰富的金属材料知识，因而顺理成章地发明了合金铅字。这种铅字比木、陶、铜材料的活字都更耐用，印刷效果也更清晰；缺点是它含有铅、锡、锑等有害金属，流毒甚广。这个故事告诉我们，在人类文明的历史上，发展和环保总是一对矛盾。

这些技术迅速推动了西方社会科技文化的发展，导致了文艺复兴。图 B.2 是 Gutenberg 在 1455 年印制的《圣经》[2]。

[1] 原藏于敦煌莫高窟，1907 年被英国考古学家 Marc A. Stein (1862–1943) 盗走，现存于大英博物馆。

[2] 图中拷贝现存于美国国会图书馆。

图 B.1: 《金刚经》，雕版，868

图 B.2: Gutenberg Bible, letterpress, 1455

19 世纪末出现了柔性版印刷，它使用柔软的橡胶或聚合物材料作印刷版，可以使用几乎所有印刷材料，包括塑料、金属薄膜、玻璃纸等特殊材料。柔性版常用于印刷产品包装。

B.1.2 凹版印刷

凹版印刷一般用铜制印版，也有用锌、铁的，多用于版画。15 世纪中叶，雕刻凹版出现于德国和意大利，它用刻刀在印版上刻画比较直的线条。针刻也是 15 世纪发源于德国，它用带有针状尖端的工具在印版上刻画。图 B.3 是德国艺术家 Albrecht Dürer (1471–1528) 的雕刻作品《圣克里斯托弗》[3]。图 B.4 是德国印象派画家 Lesser Ury (1861–1931) 的针刻作品《咖啡馆中的女人》。

图 B.3: St. Christopher, engraving by Dürer, 1521

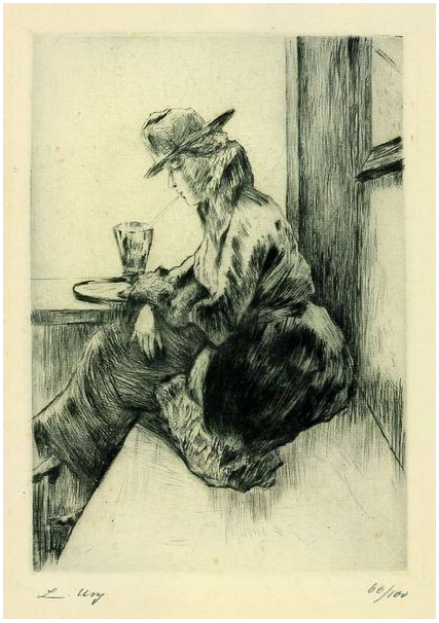


图 B.4: Woman in Cafe, drypoint by Lesser Ury

15 世纪末，德国艺术家 Daniel Hopfer (1470–1536) 发明了蚀刻法。先用针在印版上刻画，再用强酸腐蚀印版。图 B.5是 Hopfer 的蚀刻作品《士兵和他的妻子》。

[3]克里斯托弗是天主教和东正教的圣徒，传说身高六米，面目狰狞。某日他驼一小儿渡河，及其半渡，河水暴涨，小儿体沉似铅。至彼岸，克氏抱怨：未料汝体重至斯，吾几丧小儿胯下。那小儿正色道：彼时尔肩上是整个世界及其造物主，朕乃自有永有万王之王的耶稣基督。话毕，消弥于无形。

1642 年，德国艺术家 Ludwig von Siegen (1609–1680) 发明了磨刻。先用特制的工具在印版上刮磨出无数微细芒刺，再用刮刀把芒刺作不同程度的刮除，从而获得不同明暗的色调。图 B.6 是一幅 19 世纪的磨刻仿制作品《乔治亚娜 · 卡文迪许，德文郡公爵夫人》，原作是英国肖像画家 Thomas Gainsborough (1727–1788) 的同名油画。

图 B.5: The Soldier and his Wife, etching by Hopfer, 1500

图 B.6: Georgiana Cavendish, Duchess of Devonshire, mezzotint

细点蚀刻则在印版上洒一层薄薄的树脂粉，加热使之把粘附于版面。酸在树脂微粒间渗入，把版面腐蚀出大量小坑，坑中装墨。这样印制的版画有水粉画的效果，其色调纹理取决于酸的强度和腐蚀时间。

针刻和细点蚀刻可以结合使用。图 B.7 是西班牙浪漫主义画家 Francisco Goya (1746–1828) 的雕刻、针刻和细点蚀刻结合作品《理性沉睡，心魔生焉》。图 B.8 是美国画家 Mary S. Cassatt (1844–1926) 的针刻和细点蚀刻结合作品《浴室》。

B.1.3 平版印刷

凸版和凹版印刷的压印都是暴力的物理过程，所以印版日久天长会磨损。1796 年，德国演员和剧作家 Alois Senefelder (1771–1834) 发明了石版印刷。它利用油和水的互斥性，先用油基材料在光滑的石灰石 (limestone) 版上绘制印纹；再涂上阿拉伯树胶，此胶只会留在无印纹部分；然后把油墨和水的混合物涂在石版上，水附着于无印纹部分，油墨附着于印纹部分。石

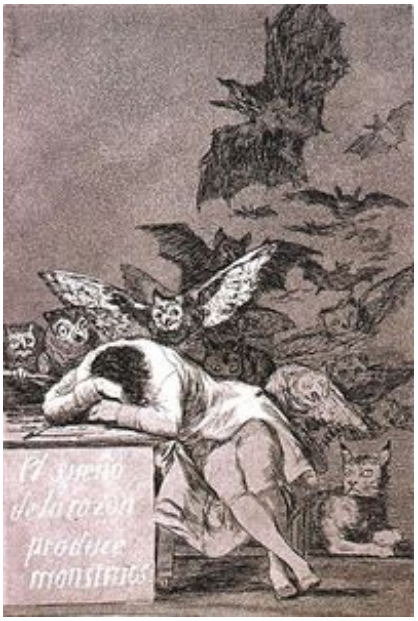


图 B.7: Sleep of Reason Produces Monsters, etching, aquatint, and drypoint by Goya, 1799

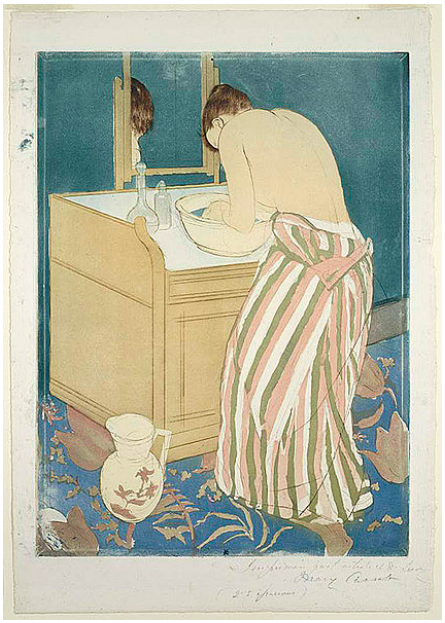


图 B.8: The Bath, drypoint and aquatint by Cassatt, 1890

版印刷利用了简单的化学原理，印版本身在印刷过程中磨损很小。

上述凸版、凹版和平版的压印过程中，印版和印刷材料都直接接触。英国作家 Robert Barclay (1648–1690) 和美国印刷工 Ira W. Rubel分别于 1875 年和 1904 年发明了胶印 (offset press) 技术。它先把印版上的图像转移 (offset) 到橡胶布 (rubber blanket) 上，再印到印刷材料上。平版和胶印结合起来的平版胶印效果更清晰，因为橡胶布和印刷材料接触得更充分；印版寿命更长，因为脏活儿累活儿都让橡胶兄弟干了；大批量印刷成本低廉。

1950 年代之后，平版胶印逐渐成为应用最广泛的商业印刷技术。而凸版、凹版因为能在印刷品上产生纹路，也还保有一席之地；凸版常用于贺卡、信封、名片、请帖、版画等；凹版常用于钞票、邮票、护照等。

B.1.4 孔版印刷

早在八世纪，勤劳勇敢的中国人民就在印刷过程中使用了镂空方法。它常用于印制纺织品或指示牌上重复的符号、标志、形状、图案。从镂空发展而来的丝网印刷则出现于宋代。丝网印刷可以使用十分广泛的印刷材料，常用于纺织品、产品标签、气球、印刷电路板、医疗设备等。

B.2 传统排版

B.2.1 手工排版

Gutenberg 在 1450 年就发明了印刷机，然而排版还是基本靠手。西文只有几十个字母，加上标点符号也没多少。早年间金属活字装在上下两个带格子的盒子 (job case) 里，大写字母放在上面的盒子 (upper case) 里，小写字母放在 (lower case) 。活字在英文里是 type，排版就是 typeset，排版工人则是 typesetter。排版工人用单个字母排成单词，单词排成行，很多行捆在一起就是一页的印模 (forme) ；印模装到印刷机上。

中文比较麻烦，常用汉字有几千个，光是装活字就得要一个很大的盘子。我猜这也是近代中国科技落后的一个原因，你想，印刷排版落后书籍就少，书籍少人们的文化知识就少。

元大德年间，宣州旌德县令王祯 (1271–1368) 发明了转轮排字盘。用轻质木材作成一个大轮盘，直径约七尺，轮轴高三尺，轮盘装在轮轴上可以自由转动。把木活字按古代韵书的分类法，分别放入盘内一个个格子里。他做了两副这样的大轮盘，排字工人坐在两副轮盘之间，转动轮盘即可找字，所谓“以字就人，按韵取字”。这样既提高了排字效率，又减轻了排字工的体力劳动，是排字技术上的一个创举。

大德二年 (1298 年) ，王祯造三万余木活字，排印《旌德县志》100 部，历时一月。1313 年，著成 13 万字的《王祯农书》，主要讲述农业生产技术。术后杂录部分记载了和农业关系不大的“造活字印书法”，其中有转轮排字盘的插图 (见本书封面) 。

B.2.2 机械排版

直到 19 世纪末期，机械排版技术才姗姗来迟。排版的机械化和压印的机械化相差了 400 多年，这充分说明排版和压印的难易程度相去甚远。

1869 年，德国商人 Karl Kastenbein发明了键盘排字机，后被英国《泰晤士报》采用。1870 年，俄国排版工 Peter Kniagininski在圣彼得堡展出了他的排版机，遗憾的是无人喝彩。他最终酗酒而死。

19 世纪末期出现的铸字排版 (hot metal typesetting，hot type) 技术使得排版的效率又上了一个台阶。它用机械的方法把熔化的专用合金注入模具，铸成活字。

1884 年，人称古腾堡第二的德裔美人 Ottmar Mergenthaler (1854–1899) 发明了林诺行式铸排机 (Linotype) ，两年后成立了摩根泰勒林诺铸排机公司

(Mergenthaler Linotype Company)。它用键盘调用活字模具，将其排成一行，整体铸造；效率高，成本低，主要用于报纸印刷。

1885 年，美国公务员 Tolbert Lanston (1844–1913) 发明了莫诺单字铸排机 (Monotype)，两年后成立了兰斯顿莫诺铸排机公司 (Lanston Monotype Machine Company)。它用键盘制作穿孔纸带，以纸带驱动搜索活字模具，铸成活字，然后排列成行。单字铸排比行式铸排灵活，纸带可以重用，主要用于书籍印刷。

1897 年，匈牙利人 C. Méray-Horváth发明了远程排版 (teletypography) 技术。它用键盘或打字机发出电报信号来控制不同地点的铸排机。1930 年代，《泰晤士报》采用了类似的技术。

1914 年，国际排版机公司 (International Typesetting Machine Company) 的因特行式铸排机 (Intertype) 上市。林诺机部件多是铸铁材料，而因特机一般用钢和铝。从此排版机领域三足鼎立。

1936 年莫诺公司 (Monotype Corp.) 在伦敦股票交易所上市；1999 年被比利时的爱克发 -吉华集团 (Agfa-Gevaert) 收购；2004 年被转卖给美国私募公司塔克 · 安东尼 (TA Associate)，并改组为莫诺图像 (Monotype Imaging)。

1990 年林诺公司与西门子的子公司地狱 (Hell) [4]合并为林诺 -地狱 (Linotype-Hell AG)，1997 年被海德堡印刷机械 (Heidelberg Printing Machines AG) 收购，其业务被拆分为字体和相关软件，以及排版两部分，前者在 2007 年被转卖给莫诺图像。

1916 年国际排版机公司被改组为因特铸排机公司 (Intertype Corp.)，1957 年与哈瑞斯公司 (Harris Corp.) 合并。1960 年代后机械排版逐渐被照相排版取代，哈瑞斯的业务遂转向通信和信息技术。

B.2.3 照相排版

从 1450 年到 19 世纪末期的 400 多年间，活版印刷在大批量文字的印刷中占据着垄断地位。在这段时期，printing 这个词和 letterpress 是同义的。图文混杂的书籍通常需要两道工序，凹版或石版印制图像，活版印制文字。

而照相排版 (phototypesetting，cold type) 技术的出现打破了这种藩篱。它利用照相感光原理在印版上制作印纹；相应的机器称为照排机 (phototypesetter)。照相排版可以用于凸版、凹版和平版。

1879 年捷克画家 Karl Klić (1841–1926) 发明了照相凹版 (photogravure)。在铜版上涂感光胶，曝光后蚀刻。照相凹版比之前的凹版技术质量高，可

[4]1929 年德国发明家 Rudolf Hell (1901–2002) 创建了该公司，1981 年被西门子收购。

以产生细腻的纹理和连续的色调。

1946 年，因特铸排的照相排版机 (Intertype Fotosetter) 上市，被美国政府印刷局 (U.S. Government Printing Office) 采用。1947 年，仙童相机仪器 (Fairchild Camera and Instrument) [5]推出了照相蚀刻机 (photo-engraving machine) 。

以上这些照排机被称为第一代照排机。后来的照排机增加了少许机械电子设备，比如键盘、穿孔纸带等，这就是第二代照排机。1954 年，法国人 René Higonnet和 Lious Moyroud发明了电子照排机 Photon 100。

B.3 数字印刷

数字电脑出现后，打印机应运而生，它们将电脑输出的数字图像转移到打印材料上。比起传统印刷技术，数字印刷成本低，应用方便，也较少浪费化学原料。主要的打印技术有：

- 光电打印机 (photoelectric printer) ，利用光电效应 (photoelectric effect) 产生静电摄取墨粉打印到纸上。按光源可以分为是激光打印机 (laser printer) 、发光二极管打印机 (LED printer) 、液晶快门打印机 (LCS printer) 等。

- 喷墨打印机 (liquid inkjet printer) ，将墨滴喷到纸张上，通过墨滴的大小来控制图像的浓淡色彩。

- 撞击式打印机 (impact printers) ，用打印头 (print head) 撞击色带 (ink ribbon) ，透过色带在纸上留下图像。包括行式打印机 (line printer)、针式打印机 (dot matrix printer) 。

- 热学打印机 (thermal printer) ，利用热学原理，包括直接加热特殊热敏纸的热敏打印机 (direct thermal printer) ，熔化蜡质墨的热蜡传递打印机 (thermal wax transfer printer) ，以及升华染料的热升华打印机 (dye-sublimation printer) 等。

- 绘图仪 (plotter) ，控制移动笔来画出矢量图。

[5]1957 年改组为仙童半导体 (Fairchild Semiconductor) 。

B.3.1 光电打印机

1930 年代，Chester F. Carlson (1906–1968) 在纽约打工时[6]，需要复制大量专利文件。他觉得光电导性也许可以用于复印：有些材料受到光照时其导电性会改变，静电可以吸附粉末，如果这些粉末是墨粉的话，就可以复制文件。

1938 年，Carlson 发明了静电复印 (xerography)，四年后获得专利。但是在他企图商业化这项技术时，几十个公司和政府部门都无动于衷，最后一家从事照相器材的小公司哈罗德 (Haloid Photographic Company) 觉得可以试一试。1959 年，哈罗德推出了第一款复印机 Xerox 914，1961 年公司改名为施乐 (Xerox)。

1969 年，施乐的 Gary K. Starkweather (1938–)[7] 在静电复印技术的基础上发明了激光打印机。激光束照射硒鼓 (selenium drum)，干墨粉就会被静电吸附。1976 年，IBM推出了第一款商用激光打印机 IBM 3800，体积庞大。第一款办公室用激光打印机是 1981 年的施乐 Star 8010，价格昂贵。1984 年惠普推出了第一款面向大众市场的激光打印机 LaserJet。

LED 打印机和 LCS 打印机[8] 分别用发光二极管 (light-emitting diode, LED) 和液晶快门 (liquid crystal shutter, LCS) 作为光源。比起一般的激光打印机，它们运动部件少，因而省电、可靠、便宜，打印速度也较快；缺点是分辨率较低。

1985 年，卡西欧推出了第一款 LCS 打印机 LCS-2400。同年，冲电气推出了 LED 打印机 OPP-6220[9]。

B.3.2 喷墨打印机

1951 年，西门子推出了第一款连续式 (continuous) 喷墨打印机。它用高压泵喷射墨滴并形成静电场，少数墨滴印到纸上，多数被回收重用。它的优点是打印速度快，喷头不会堵塞；缺点是墨水会蒸发，容易浪费。

[6]1930 年从加州理工物理系毕业后，加入贝尔实验室。1933 年大萧条时被解雇，流落到一家专利事务所。一年后跳槽到一家电池公司马洛瑞 (Mallory)，也就是后来的金霸王 (Duracell)，主持该公司专利部。1936 年始在夜校念法律，三年后获法学学士。

[7]1960 年密歇根州立大学物理学士，1966 年罗彻斯特光学硕士。1964 年加入施乐，1988 年跳槽至苹果，1997 年跳至微软。

[8]常被误称为 LCD 打印机，但是 LCD 的含义是液晶显示 (liquid crystal display)。

[9]传说 LED 打印机是卡西欧发明的，但是卡西欧网站公司历史部分只提到了 LCS 打印机；另传说冲电气 1983 年就发明了 LED 打印机，但是冲电气的三个网站都没提及，这三个网站最早的关于 LED 打印机的记录分别是 1985、1987、1989 年。

1977 年，西门子推出了可控式 (drop-on-demand) 喷墨打印机。它只在需要时才喷射，比连续式省墨、分辨率高、成本低；只是打印速度较慢。可控式的墨滴在印上纸张前需要一个加压过程，佳能、惠普、Lexmark都用加热的方法汽化墨水来增压，所以被称为热可控式 (thermal DOD) ；爱普生则用压电材料 (piezoelectric material) 把电能转化为机械能来增压，所以被称为压电可控式。压电可控式喷墨打印机因为压电材料的原因打印头成本较高，墨水成本较低，常见于商业或工业应用。

1988 年惠普推出的 DeskJet 标志着喷墨打印机走向了大众市场。目前喷墨打印机在家用市场上占据着主导地位。

B.3.3 撞击式打印机

最早的行式打印机出现于 1950 年代初期，它一次可以打出一行字，配合大型主机使用。字符装在一个鼓上的称为鼓式打印机 (drum printer) ，字符装在一条链上的称为链式打印机 (chain printer) 。链式较容易维护更换零件。1952 年的 IBM 716 是最流行的鼓式打印机，1959 年的 IBM 1403 是最流行的行式打印机。

Unix 的两个命令 `lp` 和 `lpr` 就来自行式打印机，后来其他一些操作系统也把它们的打印设备或端口命名为 `LP` 或 `LPT`，不管那打印机到底是不是行式。

针式打印机打印头上的多个针头构成一个点阵 (dot matrix) ，工作时这些针头透过色带在纸上打点组成字符，打印头从左往右移动就打出一行，打完一行后打印头复位；纸张向上移动，就能打出一页。其工作方式类似打字机，只是打字机打出的是字符。

1970 年，DEC推出了第一款针式打印机 LA30，它可以打印 80 列字符，每个字符由 5x7 的点阵构成。打印头用步进电机驱动，进纸用齿轮螺杆机构实现。

1980 年的爱普生 MX-80 发起了针式打印机对个人电脑市场的冲锋。1980 年代中期，爱普生的 LQ 系列把打印头针数从增加到 24 个，能够更好地打印东亚字符，速度也更快。

针式打印机的因为噪音大、速度慢、分辨率低，1990 年代之后逐渐被喷墨打印机取代，但是仍然保留在少数场合，比如收银机、票据机、自动柜员机等。

B.3.4 热学打印机

1970 年代出现的热敏打印技术，将特殊热敏纸张加热以获取图像，常用于传真机。1976 冲电气推出了第一款基于热敏打印的传真机 OKIFAX 7100。直到 1990 年代，多数传真机还在使用热敏打印，后被热蜡打印取代。热敏打印也常见于一些医疗器械，比如超声波仪。它的一个缺点是时间长了字迹会变淡。

热蜡转移打印机将熔化的蜡质墨喷到印鼓上，然后印到纸上，常用于标签、条形码的打印。1940 年代佐藤公司发明了这项技术。1982 年，几家日本公司推出了热蜡转移打印机。

热升华打印机将染料快速加热升华到气态，然后扩散到纸上，常用于照片打印。1980 年代末期，杜邦推出了热升华打印技术，当时还十分昂贵。1995 年法戈电子 (Fargo Electronics) [10]推出了廉价的热升华打印机 FotoFun!。

B.3.5 绘图仪

绘图仪主要用于电脑辅助设计 (computer-aided design, CAD) 。早期的绘图仪输出的是矢量图，其他打印设备输出的是点阵图。制图时，画笔横向移动，纸张纵向进给，其工作方式类似数控机床。

1959 年，加州电脑技术公司 (Calcomp Technology) [11] 推出了最早的绘图仪 Calcomp 565。1981 年，惠普推出了便携式绘图仪 HP 7470。

随着喷墨打印机和激光打印机分辨率的提高，电脑内存和速度的发展，矢量图可以在电脑端完成点阵化，然后输出到绘图设备。因此笔式绘图仪逐渐被淘汰。

B.4 数字排版

1963 年，Alphanumeric推出了第一个电脑照排系统 [12] APS 2，它以 DEC PDP 8 作为前端，还配有阴极射线管 (cathode ray tube, CRT) ；后来它的前端换为 IBM 360。这种武装了电脑技术的第三代照排机很快将铸排机赶出了历史舞台。

但是这种电脑照排机只能一个字一个字地排版，效率还是比较低。

[10]2006 年被休格斯国际 (HIG Global) 收购。

[11]1986 年被洛克希德收购，1999 年解体。

[12]德国人声称 Rudolf Hell 发明的 Digiset 才是第一个电脑照排系统，据说他还发明了电视、传真机、扫描仪。

1976，莫诺公司推出了第一款激光照排机 (laser imagesetter) Monotype Lasercomp。它用光栅图像处理器 (raster image processor, RIP) 生成点阵图像，再用激光把图像曝光到胶片上，一次可以输出整页。

激光照排机被称为第四代 (也就是最后一代) 照排机[13]，它采用电脑到胶片 (computer-to-film, CTF) 的技术路线。与之对应的是电脑到印版 (computer-to-plate, CTP)，相应的设备称为直接制版机 (platesetter)。这种方法省略了胶片，直接制作印版。1988 年，Printware 公司推出了第一款直接制版机。

不管是 CTF 还是 CTP，都需要 RIP。RIP 和它前面的流程放到 1.1节中介绍，因为他们和 TEX 的关系比较紧密。

[13]按北大方正的说法，王选 (1937–2006) 院士从 1975 年就开始研发激光照排技术，算是这个领域的先知。遗憾的是文革期间在家养病十年的王老已经不够犀利，直到 1988 年才推出中国第一台激光照排机。

再版跋

还木有写完，怎么会有后记。

原版跋

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 LaTeXer。从此你生是 LaTeX 的人，死是 LaTeX 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 LaTeXer 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 LaTeX 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 XeTeX 和 LuaTeX。

索引

人物索引

Alfred V. Aho, 阿尔佛雷德 · 艾侯, 5

Robert Barclay, 罗伯特 · 巴克雷, 154
Tim J. Berners-Lee, 蒂姆 · 伯纳斯 - 李, 6
Karl Berry, 卡尔 · 白瑞, 10

David P. Carlisle, 大卫 · 卡利斯, 64
Chester F. Carlson, 切斯特 · 卡尔森, 158
Mary S. Cassatt, 玛丽 · 卡萨特, 153
Steven D. Cochran, 斯蒂文 · 寇克阮, 69

Patrick W. Daly, 派垂克 · 达利, 128
Albrecht Dürer, 阿尔布雷希特 · 丢勒, 152

David C. Evans, 大卫 · 埃文斯, 2

Robin Fairbairns, 罗宾 · 费尔贝恩, 114
Simon Fear, 西门怕怕, 108
Robert Fourer, 罗伯特 · 福瑞尔, 5
David R. Fuchs, 大卫 · 福克斯, 9

Thomas Gainsborough, 托马斯 · 艮斯博罗, 153
David M. Gay, 大卫 · 给, 5
Charles M. Geschke, 查尔斯 · 哥谢克, 3
Denis Girou, 丹尼斯 · 杰若, 85
Charles F. Goldfarb, 查尔斯 · 歌德法布, 5
Michel Goossen, 迈克尔 · 古神, 39
Francisco Goya, 弗朗西斯科 · 戈雅, 153
George Grätzer, 乔治 · 格拉兹, 43
Johannes Gutenberg, 约翰内斯 · 古腾堡, 150

Hans Hagen, 汉斯 · 哈根, 11

Benjamin Harrison, 本杰明 · 哈里森, 134
Waldemar Hellmich, 瓦尔德玛 · 海米, 134
René Higonnet, 勒内 · 希格涅特, 157
John D. Hobby, 约翰 · 爱好, 75
Taco Hoekwater, 塔克 · 毫克水, 75
Herbert Hoover, 赫伯特 · 胡佛, 134
Daniel Hopfer, 丹尼尔 · 霍卜弗, 152

Steve Jobs, 斯蒂夫 · 乔布斯, 3
Bill N. Joy, 比尔 · 乔伊, 5

Karl Kastenbein, 卡尔 · 卡斯腾本, 155
Uwe Kern, 尤伟 · 科恩, 72
Brian W. Kernighan, 布莱恩 · 科尼汉, 5
Jonathan Kew, 乔纳森 · 酷, 11
Karl Klić, 卡尔 · 克里克, 156
Peter Kniagininski, 彼得 · 可怜人斯基, 155
Donald E. Knuth, 高德纳, 7

Leslie Lamport, 莱斯利 · 兰泡特, 9
Tolbert Lanston, 托伯特 · 兰斯顿, 156
Sergey Lesenko, 瑟基 · 莱森科, 10
Silvio Levy, 斯维 · 勒维, 8
Georg C. Lichtenberg, 乔治 · 里腾堡, 133
Raymond Lorie, 雷蒙德 · 劳里, 5

Nicolas Markey, 尼古拉斯 · 马克, 127
Malcolm D. McIlroy, 马尔考姆 · 麦克罗伊, 4
C. Méray-Horváth, 莫雷 -霍瓦斯, 156
Ottmar Mergenthaler, 奥特玛 · 摩根泰勒, 155
Michael I. Shamos, 迈克尔 · 沙毛斯, 7

Frank Mittelbach, 弗兰克·米特巴赫, 9
Robert H. Morris, 罗伯特·莫里斯, 4
Robert T. Morris, 小莫里斯, 4
Brooks Moses, 布鲁克斯·摩西, 25
Edward Mosher, 爱德华·莫谢, 5
Lious Moyroud, 路易·毛绒的, 157

Martin Newell, 马丁·牛维尔, 3

Heiko Oberdiek, 黑口·奥本迪克, 130
Tobias Oetiker, 托比·欧提克, 13
Joseph F. Ossanna, 约瑟夫·奥萨纳, 4
Wilhelm Ostwald, 威廉·奥斯特瓦尔德, 133

Scott Pakin, 斯考特·培钦, 19
Oren Patashnik, 奥润·帕塔尼克, 125
Walter Porstmann, 瓦尔特·泡茨曼, 133

Sebastian P. Rahtz, 塞巴斯蒂安·拉兹, 130
Ronald Reagan, 罗纳德·里根, 134
Wayne A. Rochester, 韦恩·罗彻斯特, 111
Keith Reckdahl, 凯斯·瑞克道, 69
Brian K. Reid, 布莱恩·瑞得, 6
Dennis M. Ritchie, 丹尼斯·瑞奇, 5
Will Robertson, 威尔·罗伯特森, 39
Tomas Rokicki, 托马斯·若奇奇, 10
Ira W. Rubel, 伊雷·鲁卜, 154
Rudolf Hell, 鲁道夫·地狱, 156

Jerome H. Saltzer, 杰若米·萨尔兹, 4
Bernd Schandl, 波得·辛得, 27
Rainer Schöpf, 雨人·肖普弗, 9
Alois Senefelder, 阿劳斯·塞尼菲尔德, 153
Axel Sommerfeldt, 艾克赛·萨摩非特, 123
Michael D. Spivak, 迈克尔·斯匹维克, 9
Robert F. Sproull, 罗伯特·斯普罗, 2
Richard M. Stallman, 理查德·斯托曼, 7
Gary K. Starkweather, 盖瑞·斯塔克维, 158
Marc A. Stein, 马克·斯坦因, 150
Ivan E. Sutherland, 伊凡·苏泽兰, 2

Till Tantau, 提·谭陶, 95
Philip Taylor, 菲利普·泰勒, 10

Lesser Ury, 莱瑟·尤里, 152

Piet van Oostrum, 皮特·范·奥斯图姆, 139
Timothy van Zandt, 提摩西·范·赞特, 25
Ludwig von Siegen, 路德维希·冯·希根, 153
Herbert Voß, 赫伯特·沃斯, 85

John E. Warnock, 约翰·沃诺克, 2
Peter J. Weinberger, 彼得·温伯格, 5
Mark A. Wicks, 马克·维克斯, 10
Peter R. Wilson, 彼得·威尔森, 121

Jiří Zlatuška, 乔治·兹拉图斯卡, 10

吴凌云, 13
吴聪慧, 40
吴聪敏, 40
孙文昌, 40
平田俊作, Shunsaku Hirata, 10
张林波, 39

李果正, 13
李树钧, 39
梅木秀雄, Hideo Umeki, 136
毕升, 150

王祯, 155

王选, 161
翁鸿翎, 40

蔡奇伟, 40
赵珍焕, Jin-Hwan Cho, 10

陈省身, Shiing-Shen Chern, 8
韩世城, Hàn Thế Thành, 10

组织机构索引

Adobe Systems, 土坯电脑, 3
Agfa-Gevaert, 爱克发 -吉华集团, 156
Alphanumeric Inc., 字母数字公司, 160
Apple Inc., 苹果公司, 1, 158
American Telephone & Telegraph, AT&T, 美国电报电话公司, 4
BEA Systems, BEA 公司, 12
Bell Labs, 贝尔实验室, 4, 158
Bendix Corp., 本迪克斯公司, 2
Boeing, 波音, 121
British Thomson Houston, BTH, 不列颠托马森休斯顿, 121
Calcomp Technology, 加州电脑技术公司, 160
CodeSourcery, 源码巫师, 25
Digital Equipment Corp., DEC, 数字设备公司, 4, 159
DuPont, 杜邦, 160
Duracell, 金霸王, 158
Mallory, 马洛瑞, 158
Electronic Data Systems, EDS, 电子数据系统公司, 9
Elsevier, 爱思维尔出版公司, 130
Elvenkind, 精灵公司, 75
Evans & Sutherland, 埃文斯 · 苏泽兰公司, 2
Fairchild Camera and Instrument, 仙童相机仪器, 157
Fairchild Semiconductor, 仙童半导体, 157
General Electric, GE, 通用电气, 121
Google, 谷歌, 6
Hal Computer Systems, 豪电脑公司, 6
Harris Corp., 哈瑞斯公司, 156
Heidelberg Printing Machines AG, 海德堡印刷机械, 156
Hell, 地狱, 156
Fargo Electronics, 法戈电子, 160
HID Global, HID 国际, 160
Hewlett-Packard, HP, 惠普, 4, 158
International Business Machine, IBM, 国际商用机器公司, 5, 158
Interleaf, 因特里弗, 10
International Typesetting Machine Company, 国际排版机公司, 156
Intertype Corp., 因特铸排机公司, 156
Intuit, 直觉公司, 10
Kluwer Academic Publishers, 克鲁沃学术出版公司, 75
Lexmark, 利盟, 159
Linotype-Hell AG, 林诺 -地狱, 156
Mergenthaler Linotype Company, 摩根泰勒林诺铸排机公司, 156
Lockheed Corp., 洛克希德, 160
London Stock Exchange, 伦敦股票交易所, 156
Lucent Technologies, 朗讯科技, 5
Massachusetts Computer Associates, 麻省计算机同伙公司, 9
Microsoft, 微软, 4, 158
Lanston Monotype Machine Company, 兰斯顿莫诺铸排机公司, 156
Monotype Corp., 莫诺公司, 1, 156
Monotype Imaging, 莫诺图像, 156
Numerical Algorithms Group, 数字算法公司, 64
Novell, 诺维尔, 5
Oracle, 甲骨文公司, 12
O'Reilly Media, 欧莱利, 6
Pragma, 普瑞格玛, 11
Printware LLC, 打印设备公司, 161
River Valley Technologies, 河谷科技公司, 10
Siemens AG, 西门子, 156
Sun Microsystems, 太阳电脑, 2
Sutherland, Sproull and Associates, 苏泽兰 · 斯普罗及其同伙, 3

TA Associate, 塔克·安东尼, 156
Unilogic, 统一逻辑公司, 7
Wang Laboratories, 王安公司, 4
Haloid Photographic Company, 哈罗德公司, 158
Xerox, 施乐, 2, 158

University of Adelaide, 阿德雷德大学, 39
American University, 美国大学, 7
Berlin Institute of Technology, 柏林工业大学, 95
California Institute of Technology, Caltech, 加州理工学院, 2, 158
University of Cambridge, 剑桥大学, 11
Carnegie Mellon University, 卡耐基梅隆大学, 2
Case Institute of Technology, 凯斯工学院, 7
Clemson University, 克莱姆森大学, 27
Columbia College, 哥伦比亚学院, 5
Cornell University, 康奈尔大学, 4
Catholic University of America, 美国天主教大学, 121
Duquesne University, 杜肯大学, 7
Eötvös Loránd University, 厄特沃什·罗兰大学, 43
Swiss Federal Institute of Technology Zurich, ETH, 苏黎世联邦理工学院, 13
Distance University of Hagen, 哈根函授大学, 39
Harvard University, 哈佛大学, 2
University of Illonois, UIUC, 伊利诺伊大学, 19
European Institute of Business Administration, INSEAD, 欧洲工商管理学院, 25
Johannes Gutenberg University, 约翰内斯·古腾堡大学, 9
Kaiserslautern University of Technology, 凯撒斯劳腾工业大学, 27
Kettering University, 凯特林大学, 10
University of Konstanz, 康斯坦茨大学, 39
University of London, 伦敦大学, 10
University of Lübeck, 吕贝克大学, 95
University of Manchester, 曼彻斯特大学, 64
University of Manitoba, 曼尼托巴大学, 43
University of Maryland, 马里兰大学, 6
Masaryk University, 捷克马萨里克大学, 10
University of Massachusetts, 马萨诸塞大学, 10
University of Michigan, 密歇根大学, 5
Michigan State University, 密歇根州立大学, 158
University of Minnesota, 明尼苏达大学, 8
Massachusetts Institute of Technology, MIT, 麻省理工学院, 2
New Jersey Institute of Technology, 新泽西理工学院, 40
Nottingham University, 诺丁汉大学, 121
Kantonsschule Olten, 奥腾大学, 13
University of Orléans，奥尔良大学, 127
University of Oxford, 牛津大学, 6
Paris Diderot University, 巴黎第七大学, 127
University of Pennsylvania, 宾州大学, 25
Princeton University, 普林斯顿大学, 5
University of Rochester, 罗彻斯特大学, 40
Rensselaer Polytechnic Institute, RPI, 伦斯勒理工学院, 121
University of Southampton, 6
Stanford University, 斯坦福大学, 2
Texas A&M University, 得克萨斯农

机大学, 10
University of Tartu, 塔图大学, 133
University of Toronto, 多伦多大学, 5
University of California, Berkeley, 加州大学伯克利分校, 5
University of Utah, 犹他大学, 2
Vassar College, 瓦沙学院, 7
Virginia Polytechnic Institute and State University, VT, 弗吉尼亚理工学院, 25
Free University of Brussels, 布鲁塞尔自由大学, 39
Wayne State Univesrity, 韦恩州立大学, 4
University of Würzburg, 维尔茨堡大学, 72
Xavier University, 泽维尔大学, 3
Yale University, 耶鲁大学, 7

Defense Advanced Research Projects Agency, DARPA, 国防部高等研究计划局, 3
General Post Office, GPO, 英国邮政总局, 10
U.S. Government Printing Office, 美国政府印刷局, GPO, 157
Los Alamos National Laboratory, 洛斯阿莫斯国家实验室, 19
National Institute of Standards and Technology, NIST, 国家标准技术研究所, 121
National Security Agency, NSA, 国家安全局, 2

American Mathematical Society, AMS, 美国数学学会, 9
American National Standards Institute, ANSI, 美国国家标准协会, 33
Computer-Aided Manufacturing International, CAM-I, 计算机辅助制造国际, 121
European Organization for Nuclear Research, CERN, 欧洲核子研究中心, 6
National Centre for Scientific Research, CNSR, 法国国家科学研究中心, 85
Deutschen Instituts für Normung, DIN, 德国标准化学会, 134
European Commission, 6
Free Software Foundation, FSF, 自由软件基金会, 7
GNU, 哥奴, 5
Institute of Electrical and Electronics Engineers, 电气电子工程师学会, 134
Internet Engineering Task Force, IETF, 互联网工程任务组, 6
Institute for High Energy Physics, 俄罗斯高能物理研究所, 10
org.impa, IMPA, 巴西理论数学与应用数学研究所, 8
International Organization for Standardization, ISO, 国际标准化组织, 3
Lucas Control System, 卢卡斯研究中心, 121
Max Planck Institute for Solar System Research, 马克斯·普朗克研究所 (Max Planck Institute for Solar System Research), 128
Mathematical Sciences Research Institute, MSRI, 数学科学研究所, 8
SIL International, SIL 国际, 11
Stanford Research Institute, SRI, 斯坦福研究所, 9
TeX User Groups, TeX 用户组, 10
World Wide Web Consortium, W3C, 万维网联盟, 6

Utrecht University, 乌德勒支大学, 139

东芝株式会社, Toshiba, 136
北大方正, 161

南开大学, Nankai University, 40
嘉南药理科技大学, Chia Nan University of Pharmacy & Science, 40
国立台湾大学, National Taiwan University, 40
武汉大学, Wuhan University, 13
西安交通大学, Xi'an Jiaotong University, 39
静宜大学, Providence University, 40
香港城市大学, City University of Hong Kong, 39
香港大学, University of Hong Kong, 7
香港理工大学, Hong Kong Polytechnic University, 39
香港科技大学, Hong Kong University of Science & Technology, 13
中国科学院, Chinese Academy of Sciences, CAS, 13
佐藤工业株式会社, SATO Corp., 160
佳能株式会社, Canon Inc., 159

冲电气工业株式会社, Oki Electric Industry, 158
卡西欧计算机株式会社, Casio Computer, 158

精工爱普生株式会社, Seiko Epson, 159

胡志明市师范大学, Ho Chi Minh City University of Pedagogy, 10

中文：11pt Adobe Song Std
英文：11pt Adobe Garamond Pro

雷坛巨擘
扛鼎力作

LaTeX Notes

雷太赫排版系统简介

第二版 v2.0

包太雷

2010 年 3 月