福昕PDF编辑器 个人版

• 永久 • 轻巧 • 自由

立即下载

购买会员

永久使用

无限制使用次数

极速轻巧

超低资源占用，告别卡顿慢

自由编辑

享受Word一样的编辑自由

扫一扫，关注公众号

http://edit.foxitreader.cn

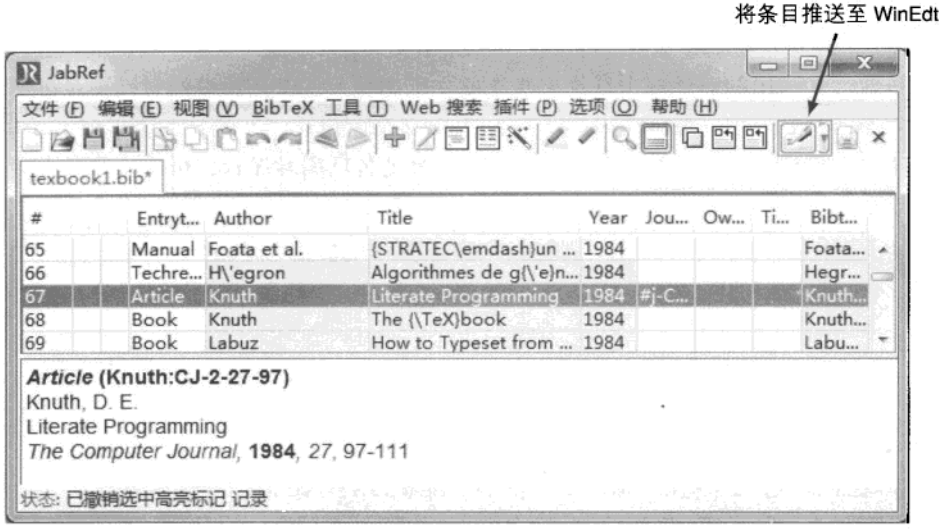


图 3.10　使用 JabRef 查看数据库文件并推送已有的文献记录

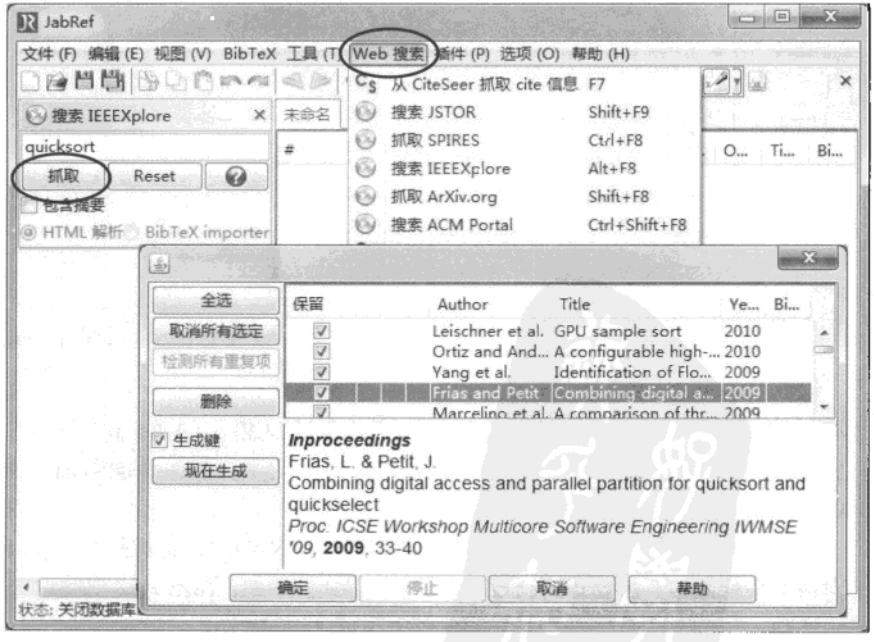


图 3.11　使用 JabRef 在线搜索抓取文献

手工建立记录条目也是需要的，在“BibTeX”菜单中可以选择一种文献记录类型直接建立，或按快捷键建立；也可以打开“BibTeX”菜单中的“新建记录向导”或点击加号按钮，在弹出的对话框中选择要建立的文献类型。JabRef 预设的文献记录类型比 3.3.1 节列举的还要多一些，可以用于扩展的 .bst 文献格式。

建立文献记录之后，会自动在下方弹出文献的各项属性设置，可以按照要求逐项填写。对于已有的记录，也可以双击打开，按同样的方式编辑（见图 3.12）。

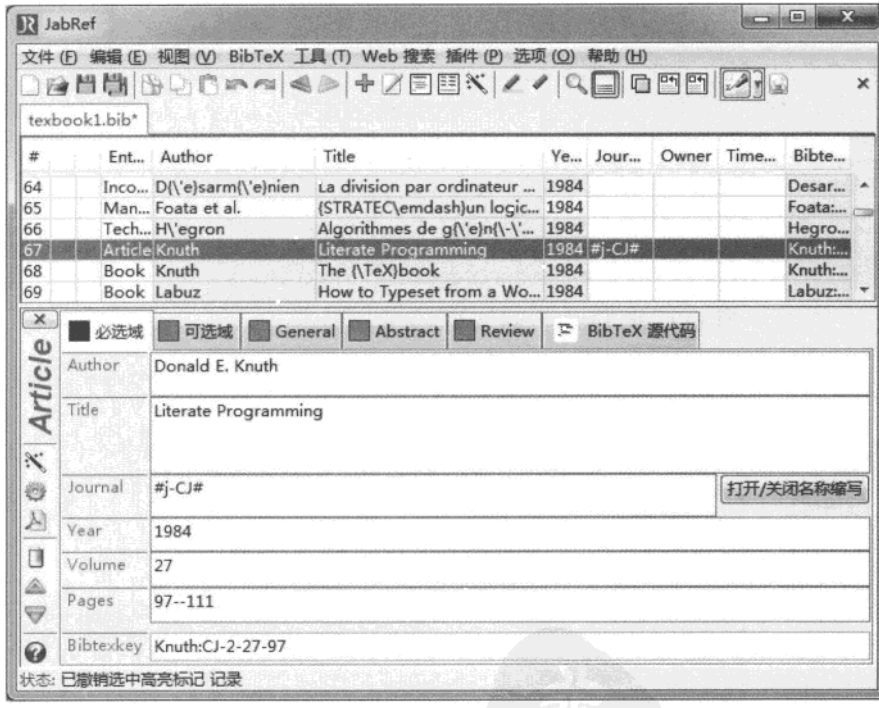


图 3.12 使用 JabRef 编辑文献记录

如果使用 Google Scholar 之类的来源，得到的是本文格式的 BibTeX 条目，可以将它们直接保存为 .bib 文件，也可以使用 JabRef 新建记录，然后在编辑记录窗口的“BibTeX 源代码”一栏直接进行编辑。

JabRef 中也可以使用在 .bib 数据库或 .bst 格式中定义的字符串宏。在 JabRef 的编辑窗口中使用宏，需要将宏写成 #宏名# 的形式。例如，在图 3.12 中使用宏 j-CJ 表示期刊“*The Computer Journal*”，在 JabRef 中写成 #j-CJ#。

在 JabRef 的“BibTeX”菜单中，可以使用“编辑导言区（preamble）”和“编辑简

写字串”命令，来手工修改 .bib 数据库中的 @preamble 项和 @string 项（参见 3.3.1 节）。

有关 JabRef 的功能大致就是这些，更详细的说明可以参见 JabRef 的联机帮助文档。本节介绍的 JabRef 只是众多文献管理软件中的一种，读者也可以自行安装其他支持 BibTeX 格式的文献管理软件。

练 习

3.4 用 JabRef 打开在练习 3.1 中找到的数据库文件 texbook1.bib，搜索标题中含有 TeX 的最早的文献。

3.5 为你书架上的学术书籍编写 BibTeX 数据库，如果不愿意一本一本手工录入，一些搜索引擎的文献导出功能可能会非常有用。

3.6 Zotero 是一款其于 Firefox 浏览器的文献管理软件，它的一大特点是可以从电子数据库的网页上批量抓取文献数据，其说明可见：

```
http://www.zotero.org/support/zh/quick_start_guide
```

如果你使用 Firefox 浏览器，试安装 Zotero，并借助它从 Google Scholar[①]、JSTOR[②]、AMS[③]、Amazon[④]或任何你喜欢的数据库中批量抓取文献数据，并导出为 BibTeX 的格式。

3.3.3 用 natbib 定制文献格式

下面来着重介绍 natbib 宏包[64] 及其所支持的文献格式。

natbib 宏包提供一种“自然”的文献引用方式，即按文献作者和年代显示文献的引用方式。它同时也提供专用于 natbib 的三种 .bst 格式：plainnat, abbrvnat 和 unsrtnat，分别对应于三种基本格式。使用 natbib 也可以产生传统的数字编号引用，同时它还提供了一些文献列表和引用的格式设置工具。

natbib 提供了许多新的引用命令，其中最常用的是 \citet 和 \citep 命令，分别对应于正文（textual）引用和带括号（parenthetical）引用。前者将产生 Knuth (1984) 这样的引用，适合作为正文的一部分，而后者的结果则类似 (Knuth, 1984)，作为括号中的补充说明，例如：

① http://scholar.google.com/
② http://www.jstor.org/
③ http://www.ams.org/joursearch/
④ http://www.amazon.com/

TEX and LATEX see Knuth (1986), Lamport (1994).

References

Donald Ervin Knuth. *The TEXbook*, volume A of *Computers & Typesetting*. Addison-Wesley, 1986.

Leslie Lamport. *LATEX: A Document Preparation System*. Addison-Wesley, Reading, Massachusett, 2nd edition, November 1994.

Frank Mittelbach and Michel Goossens. *The LATEX Companion*. Tools and Techniques for Computer Typesetting. Addison-Wesley, Boston, second edition, 2004.

图 3.13 plainnat 格式下“作者年代”形式的引用和文献列表

3-3-3

```
% 导言区
% \usepackage{natbib}
% \bibliographystyle{plainnat}
% 正文
在 \citet{lamport1994} 中提到了利用 \BibTeX{} 自动处理文献的方式，
在另一本书 \citep{mittelbach2004} 中则有进一步的格式与工具的说明。
```

在 Lamport (1994) 中提到了利用 BibTeX 自动处理文献的方式，在另一本书 (Mittelbach and Goossens, 2004) 中则有进一步的格式与工具的说明。

natbib 重定义了原来的 \cite 命令，使它在作者年代引用模式下与 \citet 等价，而是数字编号引用下与 \citep 等价，即总选择相对平常的方式。

在 natbib 中，每一种引用命令还可以带有一个 * 号，表示当文献作者有两人以上时不缩略显示，例如：

3-3-4

```
\citet{Abrahams1990} \\
\citet*{Abrahams1990}
```

Abrahams et al. (1990)
Abrahams, Berry, and Hargreaves (1990)

标准的 \cite 只能带一个可选参数，在 natbib 中引用命令（通常是 \citep，或数字引用模式的 \cite）则可以进一步带两个可选参数，分别表示引用前后增加的说明文字：

```
\citep[\S~4.3~节]{lamport1994} \\
\cite[又见][第~13~章]{mittelbach2004}
```

(Lamport, 1994, § 4.3 节)
(又见 Mittelbach and Goossens, 2004, 第 13 章)

3-3-5

引用的作者、年代和数字编号也有自己单独的命令直接得到：

```
\citeauthor{Abrahams1990} \\
\citefullauthor{Abrahams1990} \\ % 或加 *
\citeyear{Abrahams1990} \\
\citeyearpar{Abrahams1990} \\
\citenum{Abrahams1990}
```

Abrahams et al.
Abrahams, Berry, and Hargreaves
1990
(1990)
1

3-3-6

此外，命令 \citealt 和 \citealp 分别是 \citet 和 \citep 不带括号的版本，\citettext 则把任何文字看做括号中的引用，还可以用 \citetext 组合复杂的引用格式，如：

```
\citealt{Patashnik1988:btxdoc}\\
\citealp{Patashnik1988:btxdoc}\\% 有逗号
\citetext{同前} \\
\citetext{参见 \citealp{Shell2007},
  以及 \citealp{Markey2009}}
```

Patashnik 1988
Patashnik, 1988
(同前)
(参见 Shell and Hoadley, 2007, 以及 Markey, 2009)

3-3-7

natbib 新定义的引用命令还有首字母大写的形式，即 \Citet, \Citep, \Citealt, \Citealp, \Citeauthor，其输出也是强制首字母大写的。这些命令可以把文献数据库中的 von Neumann 输出为 Von Neumann。(但对这个匈牙利名字来说这种转换是错误的。)

natbib 提供了 \setcitestyle 命令来设置引用命令的输出格式，在参数中可以设置以下选项：

- 选择引用模式：authoryear 表示作者年代模式 [Daly, 2009]，numbers 表示数字序号模式 [64]，super 则表示数字上标模式 [64]；
- 括号：round 圆括号，square 方括号，或是用 open={(左括号)} 和 close={(右括号)} 来分别设置；

- 多个引用之间的标点：semicolon 分号，comma 逗号，或是用 citesep={(符号)} 来设置；
- 作者与年代之间的符号：aysep={(符号)}；
- 同一作者的几个年代间的符号：yysep={(符号)}；
- 在引用命令可选参数的说明文字前的符号：notesep={(符号)}。

通常作者年代模式多使用圆括号，数字序号引用使用方括号，几个间隔符号较少改动。natbib 宏包默认的设置相当于：

```
\setcitestyle{authoryear,round,comma,aysep={;},yysep={,},notesep={,}}
```

此外，natbib 宏包的 authoryear、numbers 和 super 选项分别对应于三种引用模式，round、square 选项分别对应于不同的引用括号，semicolon、comma 分别对应于不同的标点等。默认值是 authoryear, round, semicolon。如要换成数字编号的文献格式，可以用：

3-3-8
```
\usepackage[numbers,square]{natbib}
```

除了引用的格式，natbib 宏包也可以设置文献列表的一些排版方式。文献列表的格式主要由 .bst 格式文件决定，宏包则可以整体调整一些字体字号、间距格式等：

- \bibsection 命令决定文献列表的标题以哪种方式排版。默认是 \chapter* 或 \section*，也就是说一个文献列表就是不编号的一章或一节，可以重定义这个命令得到特别的效果。

  此外，文献列表的标题文字则由 \bibname（对 article 类是 \refname）决定，这不依赖 natbib。可以直接对其进行重定义，在 ctex 文档类中也可以用 \CTEXoptions 命令设置[58]：

3-3-9
```
\CTEXoptions[bibname={本书引用的文献}]
% 或 \renewcommand\bibname{本书引用的文献}
```

- \bibpreamble 命令的内容会在文献列表标题后，列表之前插入。如果需要对文献列表做一些说明，可以重定义这个宏，例如：

3-3-10
```
\renewcommand\bibpreamble{下面给出本文参考的几篇文章。}
```

- \bibfont 控制整个文献列表的总体字体，默认为空，如可改为小字号：

3-3-11
```
\renewcommand\bibfont{\small}
```

- \citenumfont 控制引用时编号的字体，但不影响外面括号的字体，也不影响作者年引用的格式。默认为空，可进行重定义修改。如要得到 [12] 的引用效果，可以定义：

```
\renewcommand\citenumfont{\itshape}
```
3-3-12

- \bibnumfmt 控制文献列表中编号的排版方式，但不影响引用，也不影响作者年代格式。它带有一个参数，默认值是 [#1]，可重定义修改，如使用粗体不带方括号的编号：

```
\renewcommand\bibnumfmt[1]{\textbf{#1.}}
```
3-3-13

- \bibhang 是一个长度变量，控制作者年代格式排版文献列表时，悬挂缩进的 \hangindent 值（悬挂缩进参见 2.2.1 节）。
- \bibsep 是一个长度变量，控制列表中不同文献条目之间的距离，例如取消间距：

```
\setlength{\bibsep}{0pt}
```
3-3-14

natbib 宏包可以与一些其他有关文献的宏包一起使用，或是代替一些旧的关于文献格式的宏包。重要的有：

- chapterbib 宏包可以让书籍的每一章都有单独的文献列表（通常这只用于 book 类或类似的文档类）。确切地说，它是对大型文档中每个 \include 命令导入的文件起效。natbib 可以方便地与 chapterbib 宏包 [16] 配合使用。为此，natbib 宏包提供了 sectionbib 选项，用来代替 chapterbib 宏包中的对应选项，这个选项可以使文献列表按照 \section* 的格式排版，以每章一个文献列表的符合要求。一个使用了 chapterbib 宏包的文档由多个文件组成，如下所示：

```
% main.tex
% 主文档
\documentclass{book}
\usepackage{chapterbib}
\usepackage[sectionbib]{natbib}
\begin{document}
  % ...
  \include{chap-intro}
  \include{chap-research}
  \include{chap-conclusion}
```

```
\end{document}

% chap-intro.tex
% 其中一章
\chapter{Introduction}
% ...
\bibliographystyle{plainnat}
\bibliography{foo}

% chap-research.tex
% ...
```

3-3-15

编译这样含有多个文献列表的文档，需要对每章的文档单独运行 bibtex 命令。图形界面的编辑器可能无法识别这种情况，因而可能需要配置特殊的编译命令，或直接在命令行下完成（以使用 pdfLaTeX 为例）：

```
pdflatex main
bibtex chap-intro
bibtex chap-research
bibtex chap-conclusion
pdflatex main
pdflatex main
```

- cite 宏包的功能被 natbib 宏包所代替。使用数字编号时，可以对多个引用项进行排序并压缩，即显示 [2-4,6] 这样的引用，而不是 [3,6,2,4] 的形式，为此，只要使用 sort&compress 选项：

3-3-16

```
\usepackage[numbers,sort&compress]{natbib}
```

也可以使用 sort 或 compress 选项表示只进行排序，或只压缩。

- mcite 宏包的功能被 natbib 所代替。使用 merge 选项可以把几个不同的文献放在文献列表中的同一项，为此需要使用带星号的 \citep* 命令指明主要的和次要的文献。如使用

3-3-17

```
% 导言 \usepackage[merge]{natbib}
\citep*{knuthtex1986, *lamport1994, *mittelbach2004}
```

可以将三篇文献合并，按 knuthtex1986 中的编号排版。

natbib 宏包的功能大致就是如此，一些较少使用的功能和更进一步的说明参见宏包文档 Daly [64]。

3.3.4　更多的文献格式

在 3.3.1 节中我们已经知道，只要使用 `\bibliographystyle` 命令选用不同的 BibTeX 格式文件，就可以改变文献列表的排版格式。除了表 3.4 中默认的几种，许多模板也都提供自己的格式文件，与模板一并发行。BibTeX 格式文件通常在 TeX 发行版安装目录的 `bibtex/bst` 子目录下，如在 TeX Live 的 `texmf-dist/bibtex/bst` 目录下就有 300 多个不同的格式文件，其中大部分是依附于某个机构、期刊的模板的，如 AMS、IEEE、Elsevier 的格式；也有一些是随某些有关 BibTeX 的宏包一起发布配套使用的，如 natbib 宏包附带的 plainnat 等格式；还有少量单独发布的。从许多格式中选用一种合适的 `.bst` 格式有时并不容易，不过在模板规定的情况下，应该使用模板所提供的格式。

专门为中文使用的文献格式较少。上海财经大学的吴凯编写了符合国家标准 GB/T 7714-2005 的格式文件[309]（见图 3.14），发布在 CTeX 论坛上[①]。正式在 CTAN 上发行的则有清华大学和东南大学的学位论文模板 thuthesis[315]、seuthesis[316] 所附的格式，这几种格式都是基于 natbib 宏包功能的格式，需要配合 natbib 使用。

TeX and LaTeX see [1], [2].

参考文献

[1] KNUTH D E. The TeXbook[M]. Computers & Typesetting, vol. A.[S.l.]: Addison-Wesley, 1986.

[2] LAMPORT L. LaTeX: A Document Preparation System[M]. 2nd. Reading, Massachusett: Addison-Wesley, 1994.

[3] 陈志杰, 赵书钦, 李树钧, 等. LaTeX 入门与提高 [M]. 第二版. 北京: 高等教育出版社, 2006:460.

图 3.14　符合 GB/T 7714-2005 格式的引用和文献列表

*本节内容初次阅读可略过。

① http://bbs.ctex.org/forum.php?mod=viewthread&tid=33591

使用新的文献格式时，编写 .bib 数据库文件也与 3.3.1 节所说的基本格式有一些区别，通常的格式都会增加一些文献类型和具体的项目，如电子文档的 URL，具体的用法需要参见所用格式的说明或示例。

是否还在为寻找一个合适的 BibTeX 格式而发愁？确实，尽管已经做好的格式已经不少，但如果希望达到某种自己特定的要求，找到正确的格式文件就太困难了。custom-bib 宏包正是为了解决这个问题而建立的，通过回答一系列问题，就可以按要求生成一个合适的 .bst 文件，非常实用。

custom-bib 宏包[62] 提供一个 DocStrip 程序[①]，称为 makebst。makebst 就是一个 TeX 源文件，在命令行下运行（一般不宜使用编辑器的 LaTeX 编译按钮）

```
latex makebst
```

就会出现一系列问题选项。makebst 的问题都是选择题，前面三个问题是：

> It makes up a docstrip batch job to produce a customized .bst file for running with BibTeX Do you want a description of the usage? (NO)
> \yn=**y**
>
> In the interactive dialogue that follows, you will be presented with a series of menus. In each case, one answer is the default, marked as (*), and a mere carriage-return is sufficient to select it. (If there is no * choice, then the default is the last choice.) For the other choices, a letter is indicated in brackets for selecting that option. If you select a letter not in the list, default is taken.
>
> The final output is a file containing a batch job which may be (La)TeXed to produce the desired BibTeX bibliography style file. The batch job may be edited to make minor changes, rather than running this program once again.
>
> Enter the name of the MASTER file (default=merlin.mbs)
> \mfile=
>
> Name of the final OUTPUT .bst file? (default extension=bst)
> \ofile=**bst**

这里用打字机体表示提问提示符，用粗体表示问题的回答。问题的回答用键盘输入，直接敲回车表示使用默认值。通常生成文献格式文件，前面的三个问题总是上面的回答。makebst 的问题问得非常详细，从文献作者姓名的大小写直到文献 URL 地址的编排方

① DocStrip [165] 程序是由 TeX 语言编写的程序，通常用来把许多 TeX 代码和文档一起合为一个文件，在使用时再编译生成多个文件，这里用它来完成交互式文档生成的工作。

式，大约有 100 个问题需要回答，最终将生成一个 .bst 格式文件。由于问题比较多，在生成格式文件时请保持耐心，完成整个流程可能要花不少时间。

makebst 程序运行完成，除了生成后缀为 .bst 的文献格式文件以外，还会生成一个后缀为 .dbj 的文件，它也是一个 DocStrip 程序，里面保存着所有已经回答的问题。例如，在文件 foo.dbj 中，指定文章标题格式的一条可能是：

```
%TITLE OF ARTICLE:
%   %: (def) Title plain
% tit-it,%: Title italic
% tit-qq,qt-s,%: Title and punctuation in single quotes
% tit-qq,%: Title and punctuation in double quotes
% tit-qq,qt-g,%: Title and punctuation in guillemets
% tit-qq,qt-s,qx,%: Title in single quotes
  tit-qq,qx,%: Title in double quotes
% tit-qq,qt-g,qx,%: Title in guillemets
```

上面只有 tit-qq,qx，没有被注释，也就表示文章的标题使用双引号括起来排版。可以通过加减其中的注释来修改以前拿不准或选错的问题。在完成修改以后，可以再次运行

```
latex foo.dbj
```

这样就可以重新按修改过的问题答案生成新的格式文件 foo.bst 了。

makebst 一般就使用 merlin.mbs（第二个问题的默认值）来生成格式文件，正是它定义了有关的问题和格式。关于这些问题的详细解释，可以参见 Daly [63]。

也许你还想对 .bst 格式文件本身一窥究竟。如果你打开一个 .bst 文件，会发现它也是一个纯文本的文件，里面使用一种 BibTeX 专用的语言定义了对参考文献数据库的不同项目应该如何处理，生成 TeX 代码。事实上，BibTeX 使用的 .bst 格式文件使用的是一种简单的基于栈的语言，与 PostScript 有些类似，这种语言的语法为后缀表达式形式。在 .bst 文件里面就是定义了一些描述输出格式的功能函数，例如排版文献编辑姓名的函数可能像这样：

```
FUNCTION {format.editors}
{ editor "editor" format.names duplicate$ empty$ 'skip$
    {
      "," *
      " " *
```

```
      get.bbl.editor
      *
    }
  if$
}
```

它的意思是：产生调用 format.names 函数处理 editor 域和字符串"editor"，判断结果是否为空，如果是空就跳过这一项，否则就把逗号、空格、获取的编辑名字等内容拼接为一个字符串作为输出结果。在未熟悉之前这种不常见的语法可能显得佶屈聱牙，不过即使只是做了粗略了解也可以对已经生成的 .bst 文件进行简单的修改。本书使用的文献格式就是先由 makebst 生成，再手工针对中文相关的部分修改得到的。详述 .bst 文件的语法已经超出了本书的范围，读者可参考 Markey [152]、Patashnik [194] 获得更多的信息。

练 习

3.7 查看你的 TeX 系统中安装的 .bst 文件及相关文档，或搜寻相关机构网站，看看 AMS、APS、ACM 或 IEEE 这些机构使用什么 .bst 格式文件。

3.8 使用 custom-bib 宏包的 makebst 工具，回答问题，参考本书的参考文献格式，生成一个 .bst 格式文件。

3.3.5 文献列表的底层命令

现在，让我们从自动化工具回到原始的 TeX 代码，来看看 BibTeX 究竟生成了什么东西，LaTeX 是如何处理实际的文献列表的。

如图 3.8 所示，BibTeX 处理 .aux 辅助文件后，按照 .bst 文件中预定义的格式，将会从 .bib 文件中提取文献信息，得到后缀为 .bbl 的文献列表文件。BibTeX 生成的文献列表文件其实并无神奇之处，如使用 plain 格式得到图 3.5 效果的 .bbl 文件是这样的：

```
1 \providecommand\oldacute{\'}
2 \begin{thebibliography}{1}
3
```

*本节内容初次阅读可略过。

```
\bibitem{knuthtex1986}
Donald~Ervin Knuth.
\newblock {\em The {{\TeX{}book}}}, volume~A of {\em Computers \& Typesetting}.
\newblock Addison-Wesley, 1986.

\bibitem{Lamport1994}
Leslie Lamport.
\newblock {\em {{\LaTeX}}: A Document Preparation System}.
\newblock Addison-Wesley, Reading, Massachusett, 2nd edition, November 1994.

\bibitem{mittelbach2004}
Frank Mittelbach and Michel Goossens.
\newblock {\em The {{\LaTeX}} Companion}.
\newblock Tools and Techniques for Computer Typesetting. Addison-Wesley,
  Boston, second edition, 2004.

\end{thebibliography}
```

3-3-18

BibTeX 生成的 .bbl 文件通常就包含一个 thebibliography 环境，这个环境与 enumerate、description 等列表环境颇有相似之处，就是一个带有编号的列表环境，里面的每个列表项用 \bibitem{⟨标签⟩} 命令产生。LaTeX 要的 \bibliography 命令实际要做的，除了在辅助文件中标明所用的文献数据库之外，就是检查是否已经生成了 .bbl 文献列表文件，如果已经生成就把它读入进行排版。因此，将例 3-3-18 中的代码直接写在 TeX 源文件中，也可以得到手工排版的文献列表，事实上 LaTeX 早期的许多文献列表就是这样直接手工输入的。

下面来仔细分析 thebibliography 环境。thebibliography 环境带有一个参数，这个参数是用来占位的，表示每条文献的编号宽度。当使用数字编号时，这个占位符通常就用文献列表的条目总数表示，如有 130 条文献时，数字编号的宽度自然不会超过“130”的宽度，当然使用“123”、“000”或是“abc”占位也能得到相同的效果；如果使用的不是数字编号，而是像图 3.6 中那样的字母编号，编号的最大宽度通常就需要从实际编号中选择一个占位了，如图 3.6 所对应的文献列表文件：

```
\providecommand\oldacute{\'}
\begin{thebibliography}{Lam94}

\bibitem[Knu86]{knuthtex1986}
Donald~Ervin Knuth.
\newblock {\em The {{\TeX{}book}}}, volume~A of {\em Computers \& Typesetting}.
```

```
\newblock Addison-Wesley, 1986.

\bibitem[Lam94]{Lamport1994}
Leslie Lamport.
\newblock {\em {{\LaTeX}}: A Document Preparation System}.
\newblock Addison-Wesley, Reading, Massachusett, 2nd edition, November 1994.

\bibitem[MG04]{mittelbach2004}
Frank Mittelbach and Michel Goossens.
\newblock {\em The {{\LaTeX}} Companion}.
\newblock Tools and Techniques for Computer Typesetting. Addison-Wesley,
  Boston, second edition, 2004.

\end{thebibliography}
```

3-3-19

有了例 3-3-18 和例 3-3-19 的代码，就不难看出 thebibliography 环境中其他命令的用法了。\bibitem 命令开始一条新的文献，它有一个必需的参数，是文献的被 \cite 命令所引用标签。还有一个可选的参数，给出非数字的文献编号。

\newblock 命令用来分隔文献列表中不同的“块”。BibTeX 把每条文献中内容相关的项目放在同一个块里面，不同的块之间会产生适当的间距表示分隔，\newblock 命令实际就被定义一个不大的水平间距。如果使用了文档类的 openbib 选项（参见第 138 页 2.4.1 节），那么 \newblock 命令就会被重定义为分段命令，于是参考文件列表就会被分成好多行的形式。如图 3.15 所使用的 .bbl 文件与图 3.5 完全相同，都是例 3-3-18 的代码，只是 openbib 选项不同。要得到图 3.15 中的效果，所用的代码大约是这样的：

```
\documentclass[openbib]{article}
\bibliographystyle{plain}
\begin{document}
\TeX{} and \LaTeX{} see \cite{knuthtex1986}, \cite{Lamport1994}.
\nocite{mittelbach2004}
\bibliography{tex}
\end{document}
```

3-3-20

当使用 openbib 选项时，过长的行会产生悬挂缩进的效果（见图 3.15）。悬挂缩进的距离可以用长度变量 \bibindent 控制，它在标准文档类中的默认值是 1.5 em。

使用与 natbib 宏包兼容的文献格式时，生成的 .bbl 文件会略有不同。natbib 重定义了 \bibitem 宏包，使它可以更复杂的参数。例如，本书文献列表中 Abrahams et al.

TEX and LATEX see [1], [2].

References

[1] Donald Ervin Knuth.
The TEXbook, volume A of *Computers & Typesetting*.
Addison-Wesley, 1986.

[2] Leslie Lamport.
LATEX: A Document Preparation System.
Addison-Wesley, Reading, Massachusett, 2nd edition, November 1994.

[3] Frank Mittelbach and Michel Goossens.
The LATEX Companion.
Tools and Techniques for Computer Typesetting. Addison-Wesley, Boston, second edition, 2004.

图 3.15　文档类加 openbib 选项产生的文献列表，文献格式为 plain

[1] 这一项在 .bbl 文件里面是这样的：

```
\bibitem[{Abrahams et~al.(1990)Abrahams, Berry, and Hargreaves}]
  {Abrahams1990}
\textsc{Paul~W. Abrahams}, \textsc{Karl Berry}, and \textsc{Kathryn~A.
  Hargreaves}.
\newblock \textit{{{\TeX}} for the Impatient}.
\newblock Reading, MA, USA: Addison Wesley, 1990.
\newblock ISBN 0-201-51375-7
\newline\urlprefix\url{CTAN://info/impatient/book.pdf}
```

这里，\bibitem 的可选参数可以分为三个部分：第一部分是可以省略的作者列表，第二部分是用圆括号括起来的出版年份，第三部分则是作者的完整列表。如果作者少于三个则可以没有第三部分。natbib 宏包会根据宏包的设置来从 \bibitem 的可选参数中提取信息，排版正确的文献编号和引用。

在标准文档类中，有关 thebibliography 更为底层的定义可以在 Lamport et al. [139] 中找到。不过，通常使用标准的用户命令和 natbib 的宏包的设置就可以得到足够的文献格式了。

练 习

3.9 既然 BIBTEX 已经完成了文献的排版工作，那么为什么还要了解 thebibliography 环境等底层命令的功能呢？我们在什么时候使用它们？

3.4 Makeindex 与索引

这一节我们要讨论文档的索引。索引是长篇文档的重要的组成部分，一个精心制作的索引可以大大缩短读者检索新概念的时间，可以让一本书变得更加有用。

索引的编写是创造性的工作，它需要作者根据文档的内容选取合适的关键字条目，也需要作者来决定不同索引项的编排格式。另一方面，制作索引又是一件恼人的繁复工作，它需要从篇幅庞大的文档中抽取信息、记录页码、分类排序、排版输出，因此使用自动化工具是必由之路。

3.4.1 制作索引

在 LaTeX 中制作索引，需要 .tex 源文件和外部索引程序的共同协作。在 .tex 源文件中，我们需要做以下几件事：

1. 在导言区使用 \makeindex 命令，开启索引文件输出；
2. 在导言区调用 makeidx 宏包[32]，开启索引列表排版功能；
3. 在正文中需要索引的关键字处使用 \index 命令，生成索引项；
4. 在需要生成索引的地方（通常是文档的末尾），使用 \printindex 命令，实际输出处理好的索引列表。

一个可以输出关键字索引的完整例子如下：

```
% foo.tex
\documentclass{ctexart}
\usepackage{makeidx}
\makeindex
% ...
\begin{document}
\section{勾股定理}
% 第 1 页
勾股定理在西方称为毕达哥拉斯定理（Pythagoras' theorem）。
\index{Pythagoras}
% ...
% 第 2 页
在中国常称勾股定理为商高定理。\index{商高}

```

```
15 \printindex
16 \end{document}
```

3-4-1

与参考文献类似，要生成索引需要多次编译和外部工具 Makeindex 的配合，编译带索引的文档 foo.tex 需要使用如下命令（以 XeLaTeX 编译为例）：

```
xelatex foo
makeindex foo
xelatex foo
```

这里，第一次 XeLaTeX 编译生成索引项；然后使用 makeindex 命令对索引项分类排序，生成索引列表的代码；然后在第二次 XeLaTeX 编译时引入处理后的索引列表，得到有索引列表的文档。

图 3.16　LaTeX 索引处理流程

下面着重说明 \index 命令的用法。

正如例 3-4-1 中那样，简单的索引项可以使用 \index{⟨项目⟩} 来得到，因而编译例 3-4-1 将得到如下的索引项：

索引

Pythagoras, 1

商高, 2

经过 Makeindex 程序处理过的索引表会将文档中分散的条目归并起来并排序，例如，我们如果在文档中输入如下多个索引项，将有：

第 1 页	\index{alpha}	Alpha, iv
	\index{beta}	alpha, 1, 5
第 5 页	\index{alpha}	beta, 1
	\index{gamma}	gamma, 5, iv
第 iv 页	\index{Alpha}	
	\index{gamma}	

在 \index 的参数中可以使用符号 ! 来分隔不同层次的索引项，这样将得到分级的索引项。默认支持三级列表，例如：

第 1 页	\index{language}	Chinese, 8
第 3 页	\index{language!Chinese}	history, 5
第 4 页	\index{language!Chinese!dialect}	language, 1
第 5 页	\index{language!English}	Chinese, 3, 8
	\index{history}	dialect, 4
第 8 页	\index{language!Chinese}	English, 5
	\index{Chinese}	

在 \index 的参数后面使用符号 | 则可以使用几个特殊的功能。基本的用法是使用 |(和 |) 表示页码区间，以及使用 |see{⟨条目⟩} 与 |seealso{⟨条目⟩} 表示参考条目，例如：

第 5 页	\index{alpha|(}	alpha, 5–15
第 8 页	\index{beta}	beta, 8
第 9 页	\index{Beta|see{beta}}	Beta, *see* beta
第 12 页	\index{gamma|seealso{beta}}	gamma, *see also* beta
第 15 页	\index{alpha|)}	

在 \index 的参数中使用符号 @ 可以把条目内容分成两部分，前一部分的字符串用于排序，后一部分则用来得到排版结果。通常使用这种方式来对数学公式或一些需要用 LaTeX 命令输入的特殊符号生成索引项，例如：

第 3 页	\index{n@n}	**moon**, 8
第 6 页	\index{pi@π}	n, 3
第 8 页	\index{moon@\textbf{moon}}	π, 6
第 10 页	\index{pi@\verb=\pi=}	\pi, 10

从这里也可以看出在 \index 命令的参数中可以使用各种命令，甚至是通常不能放在命令参数中的 \verb 命令（参见 2.2.5 节），这是在 LaTeX 中少有的特例。

由于符号 !、|、@ 的特殊性，如果需要在 \index 的条目中用到这些符号，需要在符号前面加引号 "，而用到引号的条目则使用两个引号 ""。\index 中的引号不影响任何 TeX 命令的使用，例如：

```
第 2 页    \index{Aha"!}                    quote("), 10
第 7 页    \index{Bob"@mail.org}            Aha!, 2
第 9 页    \index{x@$"|x"|$}                Bob@mail.org, 7
第 10 页   \index{quote(\verb=""=)}         |x|, 9
```

除了表示页码区间的 |(和 |) 是 Makeindex 内置的功能外，类似 |see 的特殊用法只是执行了一条 makeidx 宏包定义的命令 \see。LaTeX 编译程序把包含索引条目的 .tex 源文件：

```
第 9 页  \index{Beta|see{beta}}
```

编译后，会把提取出来的条目写到 .idx 辅助文件中：

```
\indexentry{Beta|see{beta}}{9}
```

而后运行的 Makeindex 程序会将上面的条目转换为 .ind 文件中的 LaTeX 代码：

```
\item Beta, \see{beta}{9}
```

最终由 \printindex 命令将 .ind 文件读入，生成索引列表的结果。Makeindex 程序并不理会符号 | 后面的内容是什么意义[①]，只是将符号 | 后面的内容加上一个反斜线 \ 和页码得到实际的页码输出方式。而在标准宏包 makeidx 中，\see 和 \seealso 只是简单地定义为：

```
\newcommand*\see[2]{\emph{\seename} #1}
\newcommand*\seealso[2]{\emph{\alsoname} #1}
\newcommand*\seename{see}
\newcommand*\alsoname{see also}
```

因而 \see{beta}{9} 就会直接输出“*see* beta”的效果，中文文档可以通过重定义命令 \seename 和 \alsoname 来对索引项汉化：

```
\renewcommand\seename{参见}
\renewcommand\alsoname{又见}
```

3-4-2

仿照 \see 和 \seealso 命令的用法，可以在索引项中使用任意的命令。例如使用 |textit 可以得到斜体的页码。也可以使用自定义的更复杂的命令，例如我们可以定义：

```
\newcommand*\numsee[2]{#2 (\emph{参见} #1) }
```

3-4-3

从而得到：

第 1 页	\index{alpha}	alpha, 1, *3*
第 3 页	\index{alpha|textit}	beta, 9 (参见 alpha)
第 9 页	\index{beta|numsee{alpha}}	

可以使用 imakeidx[21] 宏包代替 makeidx 宏包，它扩展了 \makeindex、\index 和 \printindex 的语法，提供在同一文档中生成多个索引表的功能。同时它可以在编译 TEX 文档的同时自动调用 Makeindex 程序处理不同的索引表。它的简单用法如下例所示：

```
1  \documentclass[UTF8]{ctexart}
2  \usepackage{imakeidx}
3  \makeindex[title={名词索引}]
4  \makeindex[name=persons,title={人名索引}]
5
6  \begin{document}
7  ... \index{名词}
8  ... \index[persons]{人名}
9  ...
10 \printindex           % 输出名词索引
11 \printindex[persons]  % 输出人名索引
12 \end{document}
```

3-4-4

编译使用 imakeidx 的文档需要给编译命令（如 xelatex）增加 -shell-escape（TEX Live）或 --enable-write18（MiKTEX）的命令行选项[①]，允许 TEX 编译程序调用

① 如果符号 | 后面不是小括号，Makeindex 就直接转换为命令；如果后面是小括号，则首先按小括号确定页码范围，然后把后面的内容转换为命令。

① 新版本的 MiKTEX 也支持 -shell-escape 选项。

Makeindex。可以在 TeXworks 或 WinEdt 等编辑器中统一设定。于是，只要点击两次 xelatex 命令就可以正确编译例 3-4-4 了。目前 TeX Live 2010 以后的版本默认允许直接调用 Makeindex，则可以省略命令行参数的设置。

3.4.2　定制索引格式

定制索引格式可以在两个层面上完成，一是 LaTeX 层面，通过自定义宏和重定义控制索引表的输出格式；二是在 Makeindex 工具的层面，直接控制输出的 .ind 文件。这两种方式各有其适用范围。

3.4.2.1　索引环境与格式

控制索引项和页码的格式可以在 \index 的参数中使用 @ 或 ! 符号，而这一工作可以使用自定义命令来简化。例如，如果要排版某种程序设计语言的文章，我们可以定义一个关键字的宏，它使用打字机字体输出一个关键字，同时将其加入索引表：

```
\newcommand*\keyword[1]{\texttt{#1}\index{#1@\texttt{#1}}}
```

3-4-5

实际的索引表是由 TeX 引擎读入 Makeindex 输出的 .ind 文件排版得到的。正如图 3.16 所示，编译程序从 .tex 源文件中收集索引项存入 .idx 文件，由 Makeindex 排序处理得到 .ind 文件。例 3-4-1 处理得到的 .ind 文件形如：

```
\begin{theindex}
  \item Pythagoras, 1
  \indexspace
  \item 商高, 2
  \indexspace
\end{theindex}
```

由此可以看出，与参考文献列表一样，索引最终也是通过一个 theindex 列表环境排版得到的。theindex 环境中可以使用条目命令 \item，第二、三级条目的 \subitem 和 \subsubitem 命令，以及用于分隔不同单词首字母的 \indexspace 命令。可以重定义这些环境和命令来完全控制索引列表的排版方式。

*本节内容初次阅读可略过。

不过，自行定义一个功能完善的 theindex 环境并非易事，要想方便地在 LaTeX 中自定义索引表格式，可以使用宏包的功能。在 3.4.1 节我们见过的 imakeidx 宏包[21] 除了支持自动生成多个索引表，还提供了若干索引表的格式选项。imakeidx 使用 \indexsetup{⟨选项⟩} 来设置索引的一般格式，使用 \makeindex[⟨选项⟩] 设置每个单独索引表的格式。此外，imakeidx 宏包还提供了一个 \indexprologue 命令，用来在索引前面添加一段说明文字。imakeidx 的定制功能示例如下：

```
\documentclass[UTF8]{ctexbook}
\usepackage{imakeidx}
\makeindex[%
  name=persons,              % 索引文件名（默认为 \jobname，即主文件名）
  title={人名索引},          % 索引表标题（默认为 \indexname）
  intoc=true,                % 加入目录（默认为 false，不加入目录）
  columns=2,                 % 分栏（默认为 1）
  columnsep=1cm,             % 栏间距（默认为 35pt）
  columnseprule=true,        % 分栏线（默认为 false）
  program=makeindex,         % 调用的索引程序（或用 xindy、texindy）
  options={-s mkind.ist},    % 索引程序的选项（默认为空）
  noautomatic=false          % 不自动调用索引程序（默认为 false）
]
\indexsetup{%
  level=\section*,           % 标题级别（默认 \chapter* 即不编号的章）
  toclevel=section,          % 目录级别（不带反斜线，又如 chapter）
  firstpagestyle=empty,      % 索引第一页的页面风格（默认为 plain）
  headers={人名}{人名},      % 索引的奇偶页眉
  othercode={                % 将在索引条目之前生效的任意代码
    \renewcommand{\indexspace}{\smallskip}}
}
\begin{document}
张三\index[persons]{张三} \newpage
李四\index[persons]{李四} \newpage
王五\index[persons]{王五}

\indexprologue{这里列出本文涉及的所有人名。}
```

```
\printindex[persons]
\end{document}
```

3-4-6

使用例 3-4-6 的代码，我们将大致得到如下的索引表：

人名索引

这里列出本文涉及的所有人名。

张三, 1
李四, 2
王五, 3

除了 imakeidx 之外，idxlayout 宏包也提供了一组的方便的命令来控制索引表的字体、缩进、对齐方式、分栏输出等格式，但注意要放在 imakeidx 宏包之后。有兴趣的读者可以参见宏包的手册 Titz [264]，这里就不再赘述了。

3.4.2.2　Makeindex 与格式文件

这一节我们将进一步介绍索引制作程序 Makeindex 的使用。

与大多数 TEX 相关的工具一样，Makeindex 是一个运行于命令行下的程序，它的功能是对 TEX 编译程序产生的索引项进行排序、归并，得到用于输出的索引表，其基本用法是：

```
makeindex foo.idx
```

这会得到排序整理后的 foo.ind 文件，其中文件后缀 .idx 可以省略不写。在一般的 TEX 编辑器配置中，使用 Makeindex 生成索引的按钮就是执行一条类似上面的简单命令，参数文件名与 .tex 源文件一样，然后得到需要的 .ind 文件。

不过，Makeindex 还有其他一些命令行选项，可以完成更丰富的功能。详细的命令说明可以参见 Makeindex 的 Manual Page[214]，这里只列举一些常用的选项：

*本节内容初次阅读可略过。

-o ⟨*ind*⟩ 设置输出文件为 ⟨*ind*⟩。这一项默认是输入文件的主文件名加上后缀 .ind。

-s ⟨*sty*⟩ 设置格式文件为 ⟨*sty*⟩。这一项默认为空，即按照 3.4.1 节的格式说明简单输出。格式文件一般以 .ist 为文件后缀，其语法将在后文中详细说明。

-t ⟨*log*⟩ 设置日志文件为 ⟨*log*⟩。日志文件记录了 Makeindex 处理的词条汇总信息和错误信息，默认是输入文件的主文件名加上后缀 .ilg。

例如，许多 LaTeX 宏包文档都是使用 ltxdoc 文件类或 doc 宏包排版的，它们支持一种特别的命令索引。在编译这种文档时，就需要使用专门的索引格式 gind.ist[161]。一个具体的例子是 xeCJK 宏包，它的源文件是 DocStrip 语法下的 xeCJK.dtx，编译命令就是：

```
xelatex xeCJK.dtx
makeindex -s gind.ist xeCJK
xelatex xeCJK.dtx
xelatex xeCJK.dtx
```

最令人感兴趣的无疑就是 Makeindex 用 -s 选项设定的格式文件。Makeindex 不使用格式文件也可以正常输出索引表，但使用格式文件可以更改诸如索引分组、页码输出等不同的输出格式，例如 gind.ist 就能产生类似下面这种索引：

A

alpha 9
apple 1

B

beta 3
Beta 2, 4

blue 7

由于产生的未排序索引项格式不同，像 gind.ist 这种专用的格式文件并不能随意在普通文档中使用。标准 LaTeX 并没有提供多少预定义的 .ist 格式文件。Makeindex 在发布时提供了一个 mkind.ist 格式，里面主要是对排序方式做了一些规定，忽略了大小写的区别，预定义了特殊符号和命令（如 \TeX）的排序方式，用处并不大。个别宏包和文档类为自己的索引表或词汇表定义了专门的格式文件，在这些宏包的文档中通常会专门说明其用法，格式文件本身则可以在发行版 TDS 结构的 texmf/makeindex/ 目录找到。

在多数情况下，索引的格式文件需要手工编写。格式文件的设置项目很多，大致可以分为输入格式和输出格式两大类[52、214]。不过一个简单的格式文件通常只包含几条

简单的输出设置，例如排版本书使用的格式文件如下：

```
% -*- coding: utf-8 -*-
% latexbook.ist
% 刘海洋
preamble "
\\begin{theindex}
  \\def\\seename{见}
  \\def\\alsoname{又见}
  \\providecommand*\\indexgroup[1]{\\indexspace
    \\item \\textbf{#1}\\nopagebreak}
"

postamble "\n\n\\end{theindex}\n"

group_skip "  %\n  \\indexspace\n  %\n"

headings_flag 1
heading_prefix "  %\n  \\indexgroup{"
heading_suffix "}\n  %\n"

numhead_positive "数字";
numhead_negative "数字";
symhead_positive "符号";
symhead_negative "符号";
```

3-4-7

正如例 3-4-7 所示，格式文件使用 % 作为注释符，主体是若干条关键字与值的列表。在格式文件中，可以使用双引号定界的字符串，字符串中可以有 C 语言风格的转义字符，如用 \n 表示换行符，用 \t 表示制表符。

有关输入格式的设置参见表 3.5。这些格式的默认值就是 LaTeX 的 \index 命令所使用的一般格式（参见 3.4.1 节），通常只要保持默认值就可以了。

表 3.5 Makeindex 格式文件的输入格式

关键字	类型	默认值	意义
keyword	字符串	"\\indexentry"	索引命令
arg_open	字符	'{'	参数的开定界符
arg_close	字符	'}'	参数的闭定界符
range_open	字符	'('	页码范围的开定界符
range_close	字符	')'	页码范围的开定界符
level	字符	'!'	索引项层次分隔符
actual	字符	'@'	（用于排序的）实际键标志符
encap	字符	'|'	页码和特殊命令指示符
quote	字符	'"'	引号
escape	字符	'\\'	逃逸 quote 的符号
page_compositor	字符串	"-"	复合页码分隔符

有关输出的格式参见表 3.6。

表 3.6 Makeindex 格式文件的输出格式

关键字	类型	默认值	意义
preamble	字符串	"\\begin{theindex}\n"	索引导言代码
postamble	字符串	"\n\n\\end{theindex}\n"	索引末尾代码
setpage_prefix	字符串	"\n \\setcounter{page}{"	页码设置前缀
setpage_suffix	字符串	"}\n"	页码设置后缀
group_skip	字符串	"\n\n \\indexspace\n"	组间垂直间距
headings_flag	数字	0	标明新字母（分组）的旗标
heading_prefix	字符串	""	新字母（分组）标题前缀
heading_suffix	字符串	""	新字母（分组）标题后缀
symhead_positive	字符串	"Symbols"	当旗标 headings_flag 为正数时，符号的标题
symhead_negative	字符串	"symbols"	当旗标 headings_flag 为负数时，符号的标题
numhead_positive	字符串	"Numbers"	当旗标 headings_flag 为正数时，数字的标题

续表

关键字	类型	默认值	意义
numhead_negative	字符串	"numbers"	当旗标 headings_flag 为负数时，数字的标题
item_0	字符串	"\n \\item "	第 0 级条目间分隔
item_1	字符串	"\n \\subitem "	第 1 级条目间分隔
item_2	字符串	"\n \\subsubitem "	第 2 级条目间分隔
item_01	字符串	"\n \\subitem "	第 0/1 级条目间分隔
item_x1	字符串	"\n \\subitem "	第 0/1 级条目间分隔，其中第 0 级无页码
item_12	字符串	"\n \\subsubitem "	第 1/2 级条目间分隔
item_x2	字符串	"\n \\subsubitem "	第 1/2 级条目间分隔，其中第 1 级无页码
delim_0	字符串	", "	第 0 级条目与页码分隔
delim_1	字符串	", "	第 1 级条目与页码分隔
delim_2	字符串	", "	第 2 级条目与页码分隔
delim_n	字符串	", "	多个页码的分隔
delim_r	字符串	"--"	页码范围符号
encap_prefix	字符串	"\\"	页码特殊指令前缀
encap_infix	字符串	"{"	页码特殊指令中缀
encap_suffix	字符串	"}"	页码特殊指令后缀
page_precedence	字符串	"rnaRA"	不同类型页码的次序，默认值表示小写罗马、阿拉伯数字、小写字母、大写罗马、大写字母
line_max	数字	72	最大行长度，超出长度会自动折行
indent_space	字符串	"\t\t"	自动折行的缩进
indent_length	数字	16	indent_space 的长度
suffix_2p	字符串	""	在 2 页的页码范围中代替 delim_r 和第二个页码
suffix_3p	字符串	""	在 3 页的页码范围中代替 delim_r 和后面的页码

续表

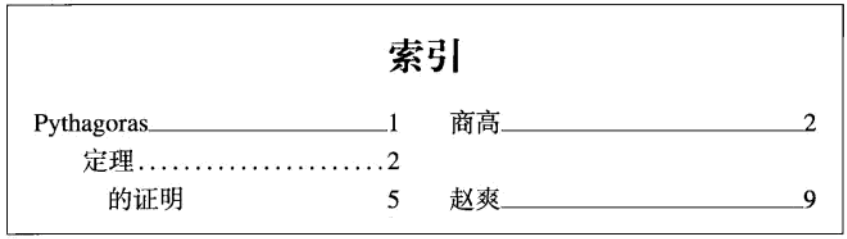
关键字	类型	默认值	意义
suffix_mp	字符串	""	在更多页的页码范围中代替 delim_r 和后面的页码

输出格式项目繁多，详细控制了从 .idx 文件到 .ind 文件的各种转换格式。其中最常用的是设置 headings_flag 为 1，并设置 heading_prefix 和 heading_suffix 的格式，这样按字母排序的索引就会有一组字母头，例 3-4-7 中就使用了这种设置。

条目与页码间的分隔也很常用，我们可以设置：

```
delim_0   "\\hrulefill"
delim_1   "\\dotfill"
delim_2   "\\hfill"
```

来得到这样效果的索引：



Makeindex 的格式文件就介绍这些，更详细的示例和说明可参见 Makeindex 的手册 Chen and Harrison [52]、Rodgers [214]。

Makeindex 程序的一个问题是它不能很好地处理中文的分组和排序。对于中文词汇较多的索引，直接使用 Makeindex 就不是很方便了。原 CCT 系统[①]提供了一个 cctmkind 程序，可以代替 Makeindex，完成在 GBK 编码下的索引表分组排序的功能。cctmkind 的使用方法与 Makeindex 大体上没有什么区别，它支持命令行选项 -C pinyin、-C stroke 和 -C mixed，分别表示按汉字拼音、按汉字笔画和按汉字音序（与英文混合）排序。目前尚没有良好支持 Unicode 编码和汉字排序的索引处理程序，当使用 XeLaTeX 等 UTF-8 编码的引擎时，可以先把 TeX 输出的 .idx 文件转换为 GBK 编码，然后用 cctmkind 处理，最后把处理的结果转换回 UTF-8 编码使用。这个过程可

① 新版本的 CCT 系统 [312] 可以在 ftp://ftp.cc.ac.cn/pub/cct/ 下载得到。

以借助 iconv[①]进行编码转换，这里不再赘述。本书的索引就是使用 cctmkind 转换编码处理的。

3.4.3　词汇表及其他

下面来看如何在 LaTeX 中生成和使用词汇表（glossary）。和索引项一样，词汇表通常同样也是使用在文档中分散给出，然后通过 Makeindex 等程序统一排序处理得到的。因此，在 3.4.1 节和 3.4.2 节介绍的索引表生成机制也同样适用于生成词汇表等其他类型的内容。

3.4.3.1　手工生成词汇表

然而，与索引表不同，标准 LaTeX 2ε 并没有给出完整的词汇表的实现，而只是提供了两个简单的命令：\makeglossary 和 \glossary。\makeglossary 命令的功能与 \makeindex 相似，用来打开词汇表文件的输出（后缀为 .glo）；而 \glossary 命令则与 \index 相似，用来生成词汇表的条目，其输入语法也与 \index 相同，实际是由 Makeindex 的格式文件定义的。利用 \makeglossary 和 \glossary 命令生成了带有词汇表条目的 .glo 文件后，可以调用 Makeindex 生成排序的词条，通常后缀为 .gls。然而，LaTeX 2ε 及标准文档类并没有定义 theindex 环境、\printindex 命令的对应物，因此，要正确读入排过序的词汇表并进行排版输出，就必须另行定义一套命令和格式。

\glossary 命令将向 .glo 文件写入以 \glossaryentry 表示的条目中（对应于索引的 \indexentry），因此，最简单的自定义 Makeindex 格式文件示例如下：

```
% simplegloss.ist
%   最简单的手工词汇表 Makeindex 格式
keyword     "\\glossaryentry"
preamble    "\\begin{theglossary}\n"
postamble   "\n\n\\end{theglossary}\n"
group_skip  ""
delim_0     "\\dotfill "
```

① iconv 是 GNU 项目的软件，用来在命令行下转换文本编码。Windows 系统的用户可以在 GNUwin32 项目 http://gnuwin32.sourceforge.net/packages/libiconv.htm 单独下载使用。

*本节内容初次阅读可略过。

而支持此格式的 TEX 源文件就可以是[①]：

```
% testgls.tex
%   最简单的手工词汇表测试
\documentclass[UTF8]{ctexart}
% 打开 \glossary 命令的写文件功能
\makeglossary
% 定义 theglossary 环境为 itemize 环境的别称
\newenvironment{theglossary}
  {\begin{itemize}}{\end{itemize}}
% 定义输出词汇表就是读文件
\newcommand\printglossary{%
  \InputIfFileExists{\jobname.gls}{\section*{词汇表}}{}}

\begin{document}
% 第 1 页
% 在正文中定义词汇表条目，前面是排序项，后面是输出格式
\glossary{LaTeX@\LaTeX：基于 \TeX{} 的文档处理系统。}
\LaTeX{} 的内容……

\newpage
% 第 2 页
\glossary{Makeindex@Makeindex：索引排序程序。}
Makeindex 的内容……

% 输出词汇表
\clearpage
\printglossary
\end{document}
```

要编译生成带词汇表的文件，可以使用下面的命令：

```
xelatex testgls
makeindex -s simplegloss.ist -o testgls.gls -t testgls.glg testgls.glo
```

① 命令 \InputIfFileExists 类似 \input 读入文件，但会先判断文件是否存在，并按情况执行代码。参见 The LaTeX3 Project [256]。

```
xelatex testgls
```

其效果是：

词汇表

- LaTeX：基于 TeX 的文档处理系统。 1
- Makeindex：索引排序程序。 2

手工从头定义一套功能完整、使用方便的词汇表并不容易，上面的例子只是一个极为简化的示例，距离实用的词汇表还有距离。我们可以把它看做是对 3.4.1、3.4.2 两节内容的进一步示例，加深对 Makeindex 机制的理解。

练 习

3.10 编译排版本节的例子，看看 TeX 生成的 .glo 文件、Makeindex 生成的 .gls 文件以及 .glg 日志文件各是什么样子。修改 .ist 文件，尝试生成其他格式的词汇表。

3.11 本节的实现仅仅是原理示意，每次都自己手工实现一个词汇表的定义并不可取。nomencl 宏包使用 Makeindex，提供了完整的术语表（nomenclature）功能，是简单术语表的很好选择。查看文档 Veytsman et al. [275] 回答以下问题：

1. nomencl 提供了哪些命令？
2. 如何使用这些命令生成术语表？试给出一个中文的例子。
3. 编译时应使用怎样的命令？

3.4.3.2　使用 glossaries 宏包

glossaries[250] 是一个功能强大的词汇表宏包，它提供了方便的接口用来在 LaTeX 中进行排版词汇表、名词和缩写管理等工作。这里对 glossaries 宏包作一简单的介绍[249]。对于追求简单的读者，也可参见练习 3.11 的 nomencl 宏包。

glossaries 提供的不仅仅是简单的条目收集、排序、输出的功能，而是真正在进行词汇条目管理的工作。因此，定义一个词条就会有引用的标签、名称、描述、复数形式等多个性质，使用词条也包括引用、复数引用、首字母大写引用、词条列表等方面。此外还可以定义专门的缩写词表等。

要使用 glossaries，首先当然是在导言区加载宏包，并使用 \makeglossaries 命令打开词汇表的功能。

定义词条是由 \newglossaryentry 命令完成的，其语法是：

\newglossaryentry{⟨标签⟩}{⟨设置⟩}

其中⟨标签⟩是词条被引用时使用的名称，而⟨设置⟩则是一组键值对，用来定义词条的具体信息。最主要的两项⟨设置⟩是表示条目内容的 name 和 description，分别是条目名和条目解释，例如：

```
\newglossaryentry{gloss}{
  name=glossary,
  description={A vocabulary with annotations for a particular subject}}
```

\newglossaryentry 也提供了不少其他可选的设置项目，例如用 plural 设置西文词条的（不是末尾加 s 的）复数形式，用 sort 设置词条用于排序的名称，用 symbol 产生有符号的词条等等。

glossaries 宏包也提供了在文档中使用词条的方式，即使用词条标签引用。引用的命令按首字母大小写和单复数形式分成四个：\gls{⟨标签⟩}、\Gls{⟨标签⟩}、\glspl{⟨标签⟩}、\Glspl{⟨标签⟩}。引用的结果是词条的 name 项，例如引用 \gls{gloss} 会得到“glossary”，而引用 \Gls{gloss} 得到“Glossary”；不过对于这种不规则复数形式的条目，必须在定义词条时就设置 plural=glossaries，引用 \glspl{gloss}、\Glspl{gloss} 才能得到正确的“glossaries”和“Glossaries”。

最后，最重要的是使用 \printglossaries 命令输出词汇表。当然，列表需要在 Makeindex 等程序排序处理后才能正确输出。

编译带词汇表的文档与编译带索引的文档类似，只不过需要把 makeindex 命令替换成 makeglossaries 命令[①]。

一个完整的带有词汇表的文件示例如下：

```
\documentclass[UTF8]{ctexart}
\usepackage{glossaries}
\makeglossaries
\begin{document}

\newglossaryentry{gloss}{
  name=glossary,
```

① makeglossaries 命令实际是调用一个 Perl 脚本，这在大部分 Linux 系统上都是没有问题的，Windows 版本的 TeX Live 也自带 Perl 解释器。但如果使用 CTEX 套装或其他基于 MiKTEX 的发行版，则需要单独安装 Perl 语言解释器，如 ActivePerl（http://www.activestate.com/activeperl）。

```
  description={A vocabulary with annotations for a particular subject},
  plural=glossaries}

\Glspl{gloss} are important for technical documents.

\newglossaryentry{sec}{
  name=分节,
  description={把文章分成章节}}
\gls{sec}对于长文档非常重要。

\printglossaries
\end{document}
```

3-4-8

使用 xelatex + makeglossaries + xelatex 的方式编译例 3-4-8，可以得到类似图 3.17的结果。

Glossaries are important for technical documents.
分节对于长文档非常重要。

Glossary

glossary A vocabulary with annotations for a particular subject.

分节 把文章分成章节.

图 3.17　例 3-4-8 的大致编译效果

例 3-4-8 只是非常简单的示例，还不可能表现出 glossaries 的全貌。但与 3.4.3.1 节相比，我们已经可以看出其功能强、使用方便的优点了。有关 glossaries 宏包的进一步功能和格式设置，这里不再详述，读者可参见文档 Talbot [250] 获得更多的信息。

练 习

3.12 在图 3.17 中显示的词汇表标题是英文的，查看文档 Talbot [250]，如何修改例 3-4-8 输出中文的“词汇表”标题？

当 TEX 遭遇 Lua

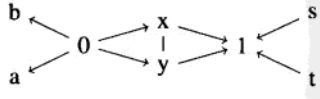
TEX 最初是作为排版工具，而不是计算机程序语言来设计实现的，因此虽然 TEX 的宏语言（参见 8.1 节）可以完成一些计算工作，但对于高效实现数值计算、组织复杂的数据结构与算法，TEX 本身的语言就力不从心了。在 LATEX 中，BIBTEX 和 Makeindex 这类外部工具的使用，也正是为了弥补 TEX 在这方面的不足。

不过，使用外部工具对 TEX 的功能进行加强，对于部分与排版内容结合紧密的工作（例如计算插图时的坐标变换计算），就不大方便了。要解决这种问题，大致有两种思路：一是通过脚本语言调用 TEX，让 TEX 能方便地利用脚本语言的各种功能，旧版本的 ConTEXt MKII 就使用这种思路把 TEX 与 Ruby 语言连接起来；另一种则是直接在 TEX 中嵌入一种计算机语言，让 TEX 与编程语言可以直接交互。采用后一种思路的就是 LuaTEX，它将 TEX 与脚本语言 Lua 结合在一起，以达成更为紧密的连接，新版本的 ConTEXt MKIV 就利用了这一特性。

除了嵌入 Lua 语言的提供的编程能力，LuaTEX 本身也是一种基于 Unicode 编码、支持 OpenType 字体的新式 TEX 引擎，可以提供现代的排版支持。fontspec、unicode-math 等字体包都同时支持 XETEX 与 LuaTEX，日本的 luatexja[150] 则对其 CJK 排版能力做了扩展。由于可以利用 Lua 语言直接修改 TEX 底层的断行、标点处理程序，luatexja 也为更为复杂的 CJK 排版提供了可能。

目前在 LuaLATEX 中，已有的 Lua 应用还不很多，但已能从中窥见在它其他引擎中不具备的能力。开发版本的 pgf 宏包就是一例，它可以调用 Lua 语言高效地进行浮点运算，绘图复杂的函数图形；还可以利用 Lua 程序实现的图可视化算法，方便地画出原来要用 Graphviz 等专门软件才能画出图论图形，例如：

```
% \usepackage{tikz} % CVS 开发版本，lualatex 编译
% \usetikzlibrary{graphs, graphdrawing, graphdrawing.force}
\tikz \graph [spring layout, orient=0-1]
  {0 -> {x,y,a,b}, {x,y,s,t} -> 1, x -- y};
```



luaindex 宏包[131] 使用 Lua 语言在 TEX 内部实现了 Makeindex 格式索引生成与排序。而 luacode 宏包[199] 则提供了在 LATEX 中方便地执行 Lua 语句的命令与环境。可以预见，随着 LuaTEX 更广泛地被使用，未来会有更多的扩展功能会使用 Lua 语言来完成，TEX 的自动化能力也会越来越强。

本章注记

自动化工具是 LATEX 最能体现其优势的地方：计数器、宏定义、辅助文件、外部工具在这里结合，LATEX 的可扩展性一显无疑。不过，LATEX 的自动化工具多而复杂，也很难找到完整的集中论述，要完全用好它亦非易事。Mittelbach and Goossens [166] 的相关章节仍然是有关工具的最佳知识来源，在本书写作中，作者也从中受益不少。

目录是基本的文档自动化工具，tocloft、titletoc 和 minitoc 是 LATEX 中有关目录最为重要的几个工具。在更为复杂的专业文档工具 memoir[294] 和 koma-script[133] 中，对于目录都有专门的一组工具进行定制，memoir 重新实现了 tocloft 的功能，而 koma-script 则附带了 tocbasic 宏包来处理目录格式。

有关交叉引用的工具也非常多，例如 cleveref[59] 提供了比 nameref 更丰富也更为智能的引用格式，zref[186] 可以引用比普通交叉引用多得多的内容（例如变量内容、文本当前的位置）等功能，并能代替 nameref、lastpage、xr 等多个宏包的功能，refstyle[70] 也提供了丰富的引用接口，限于篇幅，在书中不再一一介绍。

文献和索引生成都是借助外部工具实现自动化处理的例子，因而它们不仅有丰富的 LATEX 宏资源，也有许多外部工具可供选择。

程序方面，原始的 BIBTEX 程序不能直接支持 Unicode，排序也是针对英文字母的，`bibtexu` 就提供了更好的多语言支持（但不包括中文）。

宏包方面，bibunits[97] 提供了比 chapterbib 更完善的多文献表机制，有需要的读者不妨一试。更引人注目的是 biblatex[145] 宏包，它以一个单独的复杂 `.bst` 格式代替所有格式，然后转而利用宏包，用 LATEX 代码定制输出格式，同时也实现了有关文献引用和输出的更多相关功能（如引用文章标题），是一套强大的全新 LATEX 文献处理方案。biblatex 还可以使用新的 biber 程序代替 BIBTEX 程序，此时可以支持中文按拼音和笔画排序。biblatex 目前发展迅速，已经有越来越多的期刊开始提供基于 biblatex 的文献格式，并开始有专门为中文设计的格式（如配合 pkuthss[274] 模板使用的 caspervector[273]），本书限于篇幅不再做介绍，但追求新鲜的读者一定不能错过。

Makeindex 的重要代替品是 xindy[119]，它有更好的多语言支持（但不支持中文）和更丰富的格式配置（基于 Lisp 语言）。

LuaTEX 的发展使得在 TEX 文档中使用 Lua 脚本来完成自动化工作成为可能，在未来我们可能不再需要诸如 BIBTEX、Makeindex 这样的外部工具了。

第4章 玩转数学公式

排版数学公式是 TEX 系统设计的初衷[126]，它在 LATEX 中占有特殊的地位，也是 LATEX 最为人所称道的功能之一。或许你已经尝试过许多在电子文档中输入公式的办法而不满意，或许你使用 LATEX 就是冲着它输入公式的功能来的。不要着急，这一章，我们不仅要学会输入基本的数学公式，而且要深入下去，玩转有关公式的方方面面。

4.1 数学模式概说

TEX 有多种工作模式：输入一行文字时我们在水平模式，在水平模式下，文字、符号等各种排版元素，也就是各种盒子，都要从左到右依次水平排列；当折行分段的时候又自动进入了垂直模式，在垂直模式下，各种盒子都从上到下依次垂直排列；但最吸引人的还是数学模式，在数学模式中，输入的字符都有专门的意义，盒子的排列也遵循单独的一套特殊规则，以适应结构复杂的各种数学公式。数学公式并不是简单的符号堆砌，下面有关正态分布的公式就充分体现了这一点：

$$\int_{-\infty}^{\infty} \frac{1}{\sqrt{2\pi}\sigma} \mathrm{e}^{-\frac{(x-\mu)^2}{2\sigma^2}} \mathrm{d}x = 1 \qquad (*)$$

积分号要有上下限，被积函数分式套根式、乘方套分式，数学符号的位置复杂而又一丝不乱，“瞻之在前，忽焉在后”，这正是数学模式的魅力所在。

TEX 有两种数学公式，一种是夹杂在行文段落中的公式，如 $\int f(x)\,\mathrm{d}x = 1$，一般称为行内（inline）数学公式，或正文（in-text）数学公式；另一种就是像前面 (*) 那样单独占据整行居中展示出来的，称为显示（displayed）数学公式（或行间公式、列表公

式），显示数学环境更适合表现更复杂的数学内容。两种公式使用不同的方式进入数学模式。

在 TEX 中，行内公式一般在前后单个美元符号 $...$ 表示，例如：

4-1-1

```
交换律是 $a+b=b+a$，如 $1+2=2+1=3$。
```

交换律是 $a+b=b+a$，如 $1+2=2+1=3$。

在数学模式下，符号会使用单独的字体，字母通常是倾斜的意大利体，数字和符号则是直立体。仔细看的话，数学符号之间的距离也与一般的水平模式不同：

```
不能用 a+b=b+a，1+2=2+1=3。
```

不能用 a+b=b+a，1+2=2+1=3。

因此，在排版数学公式时，即使是没有任何特殊符号的算式 1 + 1，或者简单地一个字母变量 x，也要进入数学模式，使用 $1+1$、x，而不应该用排版普通文字的方式搞成 1+1、x。

除了使用单个美元符号，在 LATEX 中还额外定义了命令格式与环境格式的方式输入行内公式，即使用命令 \(和 \) 或是 math 环境括起一个行内数学公式，如 $a+b$ 也可以写成 \(a+b\) 或是 \begin{math}a+b\end{math}。这两种形式提供了更好的错误检查，并且可以更明确地看出公式的开始与结束，也不容易混淆。但因为输入起来比较复杂，多数人更偏爱直接使用传统的 $ 表示行内数学公式[1]。

在自定义命令时，可以使用 \ensuremath 命令得到数学模式的内容。\ensuremath 命令保证其参数的内容在数学模式下，即使原本已经在数学模式也不会发生错误，如：

4-1-2

```
% 用于 ntheorem 宏包
\renewcommand\qedsymbol{\ensuremath{\Box}}
证毕符号：\qedsymbol 或 $\qedsymbol$。
```

证毕符号：□ 或 □。

显示数学公式也有多种方式输入。基本的显示公式是不带编号的，在 TEX 中可以用连续两个美元符号 $$...$$ 界定。同样，LATEX 也定义了命令形式和环境形式的输入方法，即用 \[和 \] 命令或是 displaymath 环境括起一个显示数学公式，例如：

① 另外，在 LATEX 2ε 中 \(和 \) 是脆弱命令（参见 8.1.2 节），这也是不使用它们的重要原因。如果偏爱命令格式，可以使用 fixltx2e 宏包修正这个问题。

```
交换律是
\[ a+b=b+a, \]
如
\[
  1+2=2+1=3.
\]
```

交换律是

$$a+b=b+a,$$

如

$$1+2=2+1=3.$$

4-1-3

虽然并非必须，但最好在源代码中就把单独占据一行的显示公式放在单独的行内，使代码更清晰。推荐的方式是使用 \[...\]。$$...$$ 会产生不良的间距，缺少错误检查，并且不能正确处理 fleqn 等文档选项[267]，应该避免使用，而 displaymath 环境又可能显得冗长。值得注意的是，显示公式后面如果有标点符号，应该放在数学环境内部，紧接着公式。而且因为数学模式下不能使用汉字，所以一般就使用西文的半角标点。

LaTeX 还提供了带自动编号的数学公式，可以用 equation 环境表示，公式后还可以带引用的标签，例如：

```
\begin{equation}
  a+b=b+a \label{eq:commutative}
\end{equation}
```

$$a+b=b+a \tag{4.1}$$

4-1-4

除了 equation 环境，LaTeX 及其他一些宏包还提供了更多输入显示公式的数学环境。例如，amsmath 宏包提供了带星号的 equation* 环境，功能与 displaymath 环境相同，表示不编号的显示公式，此时 \[和 \] 就成为 equation* 环境的简写。更多的则是用来输入多行显示公式的环境，我们将在 4.4 节中进一步说明。

用 LaTeX 排版数学公式，最为常用的是宏包就属 amsmath 宏包[7] 了，它是由美国数学会（AMS, American Mathematical Society）设计开发的一个 LaTeX 宏包，全面扩展了 LaTeX 的基本数学功能。由于影响力巨大，amsmath 已经成为 LaTeX 的必备宏包，几乎所有包含 LaTeX 格式的 TeX 发行版都会安装它，大部分涉及较多数学公式的文档也都会使用 amsmath 宏包的功能。amsmath 连同 amsthm 等宏包及美国数学会的文档类一起，构成 $\mathcal{A}\mathcal{M}\mathcal{S}$-LaTeX 套件[①]。

本章的内容就基于 amsmath 宏包的扩展，在本章中，总是假定在导言区已经使用了以下命令：

① 它的前身是在 Plain TeX 下运行的 $\mathcal{A}\mathcal{M}\mathcal{S}$-TeX。

```
\usepackage{amsmath}
```

amsmath 有很多功能，我们将在后面的章节看到。数学模式和普通文本模式不同，在数学模式中不仅字符的字体、间距不同，而且空格也会被忽略，汉字也不能直接用在数学模式中，就是西文文本也不能直接输入。amsmath 提供的 \text 命令就可以用来在数学公式中插入文字[①]，例如：

4-1-5

```
$\text{被减数} - \text{减数} = \text{差}$
```

被减数 − 减数 = 差

在普通的文本中使用数学公式时也应注意随时在文本模式和数学模式下转换。例如，行内数学公式中逗号等标点处不能换行，因此列举多项公式时就应该把每项放在单独的数学环境中，项与项之前用逗号和空格隔开：

4-1-6

```
已知的变量有␣$a$,␣$b$,␣$c$,␣$d$,␣$S$,␣$R$␣和␣$T$。
```

已知的变量有 a, b, c, d, S, R 和 T。

amsmath 能识别文档类的 leqno（左侧编号）、reqno（右侧编号）或 fleqn（公式固定缩进不居中）选项的功能：

(1) $$a = b \quad \texttt{leqno}$$

$$a = b \quad \texttt{reqno} \tag{2}$$

$$a = b \quad \texttt{fleqn} \tag{3}$$

此外，amsmath 也有自己的一些宏包选项，见表 4.1。

表 4.1 amsmath 宏包选项

选项	功能
centertags	（默认）编号的公式分占多行时，编号垂直居中
tbtags	编号的公式分占多行时，编号在第一行左侧（leqno 时）或最后一行右侧（reqno 时）
sumlimits	（默认）显示公式中，巨算符 $\sum, \prod$ 等的上下标在正上下方

① 不使用 amsmath 时使用文字的方法是在数学公式中使用 \mbox 等水平盒子，但这种方法产生的字体和字号往往并不正确，不建议使用。另一种不良的做法是直接在数学环境中使用正文的字体命令。

续表

选项	功能
nosumlimits	显示公式中，求和号的上下标在角标位置
intlimits	类似 sumlimits，作用于积分号 $\int$
nointlimits	（默认）与 intlimits 相反
namelimits	（默认）类似 sumlimits，作用于 lim, max 等文字算子
nonamelimits	与 namelimits 相反

除了 amsmath，也有许多其他对数学功能进行扩展的宏包，如 mathtools 宏包[110]就对 amsmath 做了进一步扩展，我们在后面的几节还会遇到这类宏包。

4.2　数学结构

数学公式不是简单的符号连接堆砌，而是特定数学结构的组合。现在回想一下你所见过的各种数学公式，看看构成公式的结构可以分为哪些种类？不同的数学结构都有哪些规则？如何用普通的字符串来表示二维的数学公式？然后我们看看 LaTeX 是如何回答这些问题的。

4.2.1　上标与下标

上标和下标是两种最常见的数学结构，它们的形式也很朴素：上标一般在原符号的右上方，下标一般在原符号的右下方，有时也在正上方和正下方。例如：

$$10^n \qquad a_i \qquad \int_D \qquad \max_i \qquad a_i^2 \qquad \sum_{i=1}^{n}$$

在 TeX 中，上标用特殊字符 ^ 表示，下标用特殊字符 _ 表示。在数学模式中，符号 ^ 和 _ 的用法差不多相当于带一个参数的命令，如 `$10^n$` 可以得到 10^n，而 `$a_i$` 可以得到 a_i。当上标和下标多于一个字符时，需要使用分组确定上下标范围，如：

```
$A_{ij} = 2^{i+j}$
```

$A_{ij} = 2^{i+j}$

4-2-1

上标和下标可以同时使用，也可以嵌套使用。同时使用上标和下标，上下标的先后次序并不重要，二者互不影响。嵌套使用上下标时，则外层一定要使用分组。例如：

4-2-2
```
$A_i^k = B^k_i$ \qquad
$K_{n_i} = K_{2^i} = 2^{n_i}
         = 2^{2^i}$ \qquad
$3^{3^{3^{\cdot^{\cdot^{\cdot^3}}}}}$
```

$A_i^k = B_i^k \qquad K_{n_i} = K_{2^i} = 2^{n_i} = 2^{2^i} \qquad 3^{3^{3^{\cdot^{\cdot^{\cdot^3}}}}}$

这里数学公式中的空格（包括单个换行）是不起实际作用的，适当的空格可以将代码分隔得好看一些。

数学公式中是撇号，就是一种特殊的上标，表示用符号 \prime（即 ′）作上标。撇号可以与下标混用，也可以连续使用（普通的上标不能连续使用），但不能与上标直接混用，如：

4-2-3
```
$a = a'$, $b_0' = b_0''$,
${c'}^2 = (c')^2$
```

$a = a', b_0' = b_0'', c'^2 = (c')^2$

类似地，LaTeX 默认的字体没有直接表示角度的符号，可以用符号 \circ（即 ∘）的上标表示，如：

4-2-4
```
$A = 90^\circ$
```

$A = 90^\circ$

或定义为一个意义明显的命令：

4-2-5
```
\newcommand\degree{^\circ}
```

在显示公式中，多数数学算子（参见 4.3.2 节）的上下标，位置是在正上或正下方，如：

4-2-6
```
\[
  \max_n f(n) = \sum_{i=0}^n A_i
\]
```

$$\max_n f(n) = \sum_{i=0}^n A_i$$

但对积分号等个别算子，显示公式中的上下标也在右上右下角：

4-2-7
```
% 导言区 \DeclareMathOperator\dif{d\!}
\[ \int_0^1 f(t) \dif t
= \iint_D g(x,y) \dif x \dif y \]
```

$$\int_0^1 f(t)\,\mathrm{d}t = \iint_D g(x,y)\,\mathrm{d}x\,\mathrm{d}y$$

不过，在行内公式中，为了避免过于拥挤或产生难看的行距，所有算子的上下标也都在角标的位置了，如 $\max_n f(n) = \sum_0^n A_i$ 将得到 $\max_n f(n) = \sum_0^n A_i$。

4.1 节介绍的 amsmath 宏包的几个选项可以控制显示公式中一般巨算符和积分号的上下标位置，除此之外，也可以手工改变上下标的位置。在上下标前面用 \limits 命令会使上下标在正上正下方，这正是通常上下限（limits）的排版方式。而使用 \nolimits 则使上下标在角上，例如：

```
\[
  \iiint\limits_D \mathrm{d}f
    = \max\nolimits_D g
\]
```

$$\iiint\limits_D \mathrm{d}f = \max\nolimits_D g$$ 4-2-8

```
$\sum\limits_{i=0}^n A_i$ 不如用
$\sum_{i=0}^n A_i$ 更适合文本段落。
```

$\sum\limits_{i=0}^n A_i$ 不如用 $\sum_{i=0}^n A_i$ 更适合文本段落。

有时需要在符号的左上、左下角加角标，此时可以在要加角标字符前面使用空的分组，给空分组加角标，如 ${}_m^n H$ 将得到 ${}_m^n H$。不过这种不标准的方法得到的效果往往不尽如人意，间距和对齐都不合理，手工调整（参见 4.5.3 节）也比较麻烦，此时可以用 mathtools 宏包的 \prescript{上标}{下标}{元素} 来处理，例如：

```
% \usepackage{mathtools}
$\prescript{n}{m}{H}_i^j < L$
```

${}_m^n H_i^j < L$ 4-2-9

给符号左边加角标并不常见，很多时候只是需要给个别算子标记，而且还不应影响算子的上下限，此时可以 amsmath 提供的 \sideset 命令，例如：

```
\[ \sideset{_a^b}{_c^d} \sum_{i=0}^n A_i
 = \sideset{}{'} \prod_k f_i \]
```

$$\sideset{_a^b}{_c^d}\sum_{i=0}^n A_i = \sideset{}{'}\prod_k f_i$$ 4-2-10

但注意 \sideset 命令仅用于排版 $\sum$、$\prod$ 等巨算符的角标，不应用在其他地方。

amsmath 还提供了 \overset 和 \underset 命令，用来给任意符号的上下方添加标记，这种命令有点像加了 \limits 的巨算符上下标：

*本节后面内容初次阅读可跳过

4-2-11

```
$\overset{*}{X}$ \qquad
$\underset{*}{X}$ \qquad
$\overset{*}{\underset{\dag}{X}}$
```

$\overset{*}{X}$ $\underset{*}{X}$ $\overset{*}{\underset{\dagger}{X}}$

TEX 中的上下标是互不影响的，因此 `$A_m^n$` 得到 A_m^n 而不是 A_m^n。可是如果真的需要排版 A_m^n 的话，也有办法，简单地处理就是把上下标加在空的分组上，不过对大小不一的符号可能位置不够精确，此时还可以使用 2.1.1.3 节提到的“幻影”（phantom）来处理：

4-2-12

```
$A_m{}^n$ 或 $A_m^{\phantom{m}n}$
```

A_m^n 或 A_m^n

这类形式特殊的上下标在数学中也的确有其用武之地，张量代数中这种记法可以大大简化求和式的书写。tensor 宏包就专门用来排版这种张量，它主要提供 \indices 和 \tensor 两个命令，\indices 用于产生连续的复杂上下标，而 \tensor 则产生带有上下标的张量，例如：

4-2-13

```
% 导言区 \usepackage{tensor}
$M\indices{^a_b^{cd}_e}$ \qquad
$\tensor[^a_b^c_d]{M}{^a_b^c_d}$
```

$M\indices{^a_b^{cd}_e}$ $\tensor[^a_b^c_d]{M}{^a_b^c_d}$

有关 tensor 宏包的其他命令和进一步设置，可参见宏包文档 Ratcliffe [202]。

也许你还想使用上下标表示化学式，比如 H_2O（水）或是 CH_3COO^-（乙酸根）。直接把它们作为数学式输入看起来十分笨拙，即写成 `H$_2$O` 和 `CH$_3$COO$^-$`，要保证正确的字体并对齐所有的上下标使问题变得复杂。事实上，一般的化学式的形式规则远比数学式要简单，因此专业的化学宏包 mhchem（这同时也是使用最为广泛的化学宏包）能将问题简化。\ce 命令用来输入化学式，并且在大多数情况下能自动判断正确的上下标，当然也可以指出必要的上下标：

4-2-14

```
% 导言区 \usepackage{mhchem}
醋中主要是 \ce{H2O}，含有 \ce{CH3COO-}。

\ce{^{227}_{90}Th} 元素具有强放射性。
```

醋中主要是 H_2O，含有 CH_3COO^-。
$^{227}_{90}Th$ 元素具有强放射性。

\ce 可以同时用在文本或数学模式中，甚至可以用它直接输入化学方程式：

```
\begin{equation}
\ce{2H2 + O2 ->[\text{燃烧}] 2H2O}
\end{equation}
```

$$2\,H_2 + O_2 \xrightarrow{\text{燃烧}} 2\,H_2O \quad (4.2)$$

4-2-15

有关化学式的更多用法，可参见 mhchem 宏包文档 Hensel [103]。

4.2.2　上下画线与花括号

\overline 和 \underline 命令可用来在公式的上方和下方划横线，例如：

```
$\overline{a+b} =
  \overline a + \overline b$ \\
$\underline a = (a_0, a_1, a_2, \dots)$
```

$$\overline{a+b} = \overline{a} + \overline{b}$$
$$\underline{a} = (a_0, a_1, a_2, \dots)$$

4-2-16

而且这种结构可以任意嵌套或与其他数学结构组合：

```
$ \overline{\underline{\underline a}
+ \overline{b}^2} - c^{\underline n} $
```

$$\overline{\underline{\underline{a}} + \overline{b}^2} - c^{\underline{n}}$$

4-2-17

amsmath 提供了在公式上下加箭头的命令，使用方法与 \overline 和 \underline 类似：

```
$\overleftarrow{a+b}$\\
$\overrightarrow{a+b}$\\
$\overleftrightarrow{a+b}$\\
$\underleftarrow{a-b}$\\
$\underrightarrow{a-b}$\\
$\underleftrightarrow{a-b}$
```

$$\overleftarrow{a+b}$$
$$\overrightarrow{a+b}$$
$$\overleftrightarrow{a+b}$$
$$\underleftarrow{a-b}$$
$$\underrightarrow{a-b}$$
$$\underleftrightarrow{a-b}$$

4-2-18

数学字母的重音标记和宽标记（参见 4.3.1 节）与这里的命令用法类似，只是通常比较短，也不能任意伸长，不过对于单个字母往往位置更准确，实际中应该按意义和效果酌情使用，如：

```
$\vec x = \overrightarrow{AB}$
```

$$\vec{x} = \overrightarrow{AB}$$

4-2-19

除了横线和箭头，数学公式上下还可以使用 \overbrace 和 \underbrace 带上花括号，如：

4-2-20
```
$\overbrace{a+b+c} = \underbrace{1+2+3}$
```
$$\overbrace{a+b+c} = \underbrace{1+2+3}$$

并且可以使用上下标在花括号上作标注：

4-2-21
```
\[ ( \overbrace{a_0,a_1,\dots,a_n}
  ^{\text{共 $n+1$ 项}} ) =
( \underbrace{0,0,\dots,0}_{n} , 1 ) \]
```
$$(\overbrace{a_0,a_1,\dots,a_n}^{\text{共 } n+1 \text{ 项}}) = (\underbrace{0,0,\dots,0}_{n},1)$$

类似地，mathtools 宏包还提供了在数学公式上下加方括号的命令：

\underbracket[⟨线宽⟩][⟨伸出高度⟩]{⟨内容⟩}
\overbracket[⟨线宽⟩][⟨伸出高度⟩]{⟨内容⟩}

4-2-22
```
\[ \underbracket{\overbracket{1+2}+3}_3 \]
```
$$\underbracket{\overbracket{1+2}+3}_{3}$$

练 习

4.1 考虑如何排版这种交错的括号：

$$a+\overbrace{b+\underbrace{c+d}_{n}+e}^{m}+f$$

4.2.3 分式

分式（fraction）也是数学公式中极为常见的结构。在 LaTeX 中，分式用 \frac⟨分子⟩⟨分母⟩ 得到，如：

4-2-23
```
\[
\frac 12 + \frac 1a = \frac{2+a}{2a}
\]
```
$$\frac{1}{2}+\frac{1}{a}=\frac{2+a}{2a}$$

在行内公式和显示公式中，分式的大小是不同的。行内公式中分子分母都用较小的字号排版，以免超出文本行高度：

```
通分计算 $\frac 12 + \frac 1a$
得 $\frac{2+a}{2a}$
```

通分计算 $\frac{1}{2}+\frac{1}{a}$ 得 $\frac{2+a}{2a}$

已经在分子或分母中的分式，也会按行内公式的大小排版：

```
\[  \frac{1}{\frac 12 (a+b)}
= \frac{2}{a+b}  \]
```

$$\frac{1}{\frac{1}{2}(a+b)}=\frac{2}{a+b}$$ 4-2-24

有时需要指定较大或较小的分式，则可以使用 amsmath 提供的 \dfrac 和 \tfrac 分别指定显示格式（display style）和正文格式（text style）的分式：

```
\[
\tfrac 12 f(x) =
  \frac{1}{\dfrac 1a + \dfrac 1b + c}
\]
```

$$\tfrac{1}{2}f(x)=\frac{1}{\dfrac{1}{a}+\dfrac{1}{b}+c}$$ 4-2-25

连分式（continued fraction）是一种特殊的分式，amsmath 提供的 \cfrac 专用于输入连分式。这个命令可以带一个可选的参数 l, c 或 r，表示左、中、右对齐，默认是居中，如

```
\[  \cfrac{1}{1+\cfrac{2}{%
      1+\cfrac{3}{1+x}}}  =
    \cfrac[r]{1}{1+\cfrac{2}{%
      1+\cfrac[l]{3}{1+x}}}   \]
```

$$\cfrac{1}{1+\cfrac{2}{1+\cfrac{3}{1+x}}}=\cfrac{1}{1+\cfrac{2}{1+\cfrac{3}{1+x}}}$$ 4-2-26

行内公式中的分式，如 $\frac{a}{b}$，在大多数情况下仍显得过于拥挤，并不好看，因此一些出版机构的格式指南要求使用在正文中使用 a/b 代替 $\frac{a}{b}$ 的形式。此时通常要注意在适当的位置加上括号，如写 $1/(a+b)$ 而不是 $1/a+b$。不过 $1/a+b$ 的形式有一点歧义，它可能理解为 $1/(a+b)$ 或是 $(1/a)+b$。一个可能的办法是对后者使用类似 $^1\!/_a+b$ 的形式来排版分子分母都很短的小分式，这是由 xfrac 宏包[111] 提供的 \sfrac 命令得到的①：

① 更陈旧一些的 nicefrac 宏包 [207] 提供了类似的分式效果，不过功能稍差。

4-2-27
```
% \usepackage{xfrac}
区别 $\sfrac 1a + b$ 和 $1/(a+b)$
```

区别 $1/a + b$ 和 $1/(a+b)$

还有一些类似分数分成上下两半的数学结构，如二项式系数 $\binom{n}{k}$。amsmath 提供了 \binom 来输入二项式系数，其用法与 \frac 类似：

4-2-28
```
\[
(a+b)^2 = \binom 20 a^2
  + \binom 21 ab + \binom 22 b^2
\]
```

$$(a+b)^2 = \binom{2}{0}a^2 + \binom{2}{1}ab + \binom{2}{2}b^2$$

与 \frac 类似，\binom 也有指定大小的形式 \tbinom 和 \dbinom，分别表示正文格式和显示格式大小的二项式系数。

其他类似分数的形式，可以由 amsmath 提供的更一般的广义分式命令

\genfrac{⟨左括号⟩}{⟨右括号⟩}{⟨线宽⟩}{⟨大小⟩}{⟨分子⟩}{⟨分母⟩}

得到。其中 ⟨线宽⟩ 和 ⟨大小⟩ 如果为空表示默认值；⟨大小⟩ 可以是 0, 1, 2, 3，分别表示 \displaystyle, \textstyle, \scriptstyle, \scriptscriptstyle 四种数学字号（参见 4.5.2 节），如：

```
\[
\genfrac{[}{]}{0pt}{}{n}{1} = (n-1)!,
\qquad n > 0.
\]
```

$$\genfrac{[}{]}{0pt}{}{n}{1} = (n-1)!, \qquad n > 0.$$

通常并不直接使用广义分式命令，而是用它来定义新的分式形式，如第一类 Stiring 数[87]：

4-2-29
```
\newcommand\stiring[2]{\genfrac{[}{]}{0pt}{}{#1}{#2}}
\newcommand\dstiring[2]{\genfrac{[}{]}{0pt}{0}{#1}{#2}}
\newcommand\tstiring[2]{\genfrac{[}{]}{0pt}{1}{#1}{#2}}
\[  \stiring{n}{1} = (n-1)!, \qquad n > 0.  \]
```

$$\stiring{n}{1} = (n-1)!, \qquad n > 0.$$

4.2.4　根式

根式是又一种常见的数学结构，在 LaTeX 中用单参数的命令 \sqrt 得到，同时可以带一个可选参数，表示开方的次数，如：

```
$\sqrt 4 = \sqrt[3]{8} = 2$
```

$\sqrt 4 = \sqrt[3]{8} = 2$　　4-2-30

嵌套使用根式或与其他数学结构结合也很常见：

```
\[
\sqrt[n]{\frac{x^2 + \sqrt 2}{x+y}}
\]
```

$$\sqrt[n]{\frac{x^2 + \sqrt 2}{x+y}}$$　　4-2-31

不过如果开方的次数不是简单的整数，或者被开方的内容过长（甚至超过一行），通常就不使用根式的形式，而通常改用等价的指数形式：

```
\[
(x^p+y^q)^{\frac{1}{1/p+1/q}}
\]
```

$$(x^p+y^q)^{\frac{1}{1/p+1/q}}$$　　4-2-32

有时可能对开方次数的排版位置不满意，可以用 amsmath 提供的 \uproot 和 \leftroot 命令进行调整，命令参数是整数，移动的单位是很小的一段距离，如（对比例 4-2-31）：

```
\[
\sqrt[\uproot{16}\leftroot{-2}n]
  {\frac{x^2 + \sqrt 2}{x+y}}
\]
```

$$\sqrt[\uproot{16}\leftroot{-2}n]{\frac{x^2 + \sqrt 2}{x+y}}$$　　4-2-33

根式的高度是随内容而改变的，但当几个根式并列时，有时需要它们有统一的高度，此时可以使用 \vphantom（参见 2.1.1.3 节）占位，如：

```
$\sqrt{\frac 12} <
  \sqrt{\vphantom{\frac12}2}$
```

$\sqrt{\frac 12} < \sqrt{\vphantom{\frac12}2}$　　4-2-34

特别地，数学支架 \mathstrut 表示有一个圆括号高度和深度的支架，它常用来平衡不同高度和深度的字母：

4-2-35

```
$\sqrt b \sqrt y$ \qquad
$\sqrt{\mathstrut b} \sqrt{\mathstrut y}$
```

$$\sqrt{b}\sqrt{y} \qquad \sqrt{b}\sqrt{y}$$

4.2.5 矩阵

最后一类数学结构是矩阵（matrix）。在基本 LaTeX 中，矩阵是用与 Plain TeX 一样的命令 \matrix 和 \pmatrix 排版的，不过它们的语法与 LaTeX 的基本语法不大一致，所以在 AMS-LaTeX 下被一系列矩阵环境所取代[67]。复杂的矩阵则是使用 array 环境类似表格一样排版的，这种方法相对灵活但使用复杂，我们将在 5.1.1 节介绍。下面主要说明较常用的方法，即使用 amsmath 提供的一系列矩阵环境排版，各种矩阵环境的区别在于外面的括号不同：

matrix 环境 $\begin{matrix} a & b \\ c & d \end{matrix}$ bmatrix 环境 $\begin{bmatrix} a & b \\ c & d \end{bmatrix}$ vmatrix 环境 $\begin{vmatrix} a & b \\ c & d \end{vmatrix}$

pmatrix 环境 $\begin{pmatrix} a & b \\ c & d \end{pmatrix}$ Bmatrix 环境 $\begin{Bmatrix} a & b \\ c & d \end{Bmatrix}$ Vmatrix 环境 $\begin{Vmatrix} a & b \\ c & d \end{Vmatrix}$

在矩阵环境中，不同的列用符号 & 分隔，行用 \\ 分隔，矩阵每列中元素居中对齐，例如：

4-2-36

```
\[  A = \begin{pmatrix}
  a_{11} & a_{12} & a_{13} \\
  0 & a_{22} & a_{23} \\
  0 & 0 & a_{33}
\end{pmatrix}  \]
```

$$A = \begin{pmatrix} a_{11} & a_{12} & a_{13} \\ 0 & a_{22} & a_{23} \\ 0 & 0 & a_{33} \end{pmatrix}$$

在矩阵中经常使用各种省略号（参见 4.3.5 节），即 \dots, \vdots, \ddots 等：

4-2-37

```
\[  A = \begin{bmatrix}
  a_{11} & \dots & a_{1n} \\
   & \ddots & \vdots \\
  0 & & a_{nn}
\end{bmatrix}_{n\times n}  \]
```

$$A = \begin{bmatrix} a_{11} & \dots & a_{1n} \\ & \ddots & \vdots \\ 0 & & a_{nn} \end{bmatrix}_{n\times n}$$

amsmath 还提供了可以跨多列的省略号 \hdotsfor{(列数)}，如：

```
\[  \begin{pmatrix}
1 & \frac 12 & \dots & \frac 1n \\
\hdotsfor{4} \\
m & \frac m2 & \dots & \frac mn
\end{pmatrix}  \]
```

$$\begin{pmatrix} 1 & \frac{1}{2} & \dots & \frac{1}{n} \\ \hdotsfor{4} \\ m & \frac{m}{2} & \dots & \frac{m}{n} \end{pmatrix}$$

4-2-38

mathdots 宏包[151] 使 \vdots 和 \ddots 在不同字号都能正常使用，还提供了反向斜省略号 \iddots（$\iddots$），方便排版一些矩阵[①]。

矩阵可以嵌套使用，如分块矩阵：

```
\[  \begin{pmatrix}
\begin{matrix} 1&0\\0&1 \end{matrix}
& \text{\Large 0} \\
\text{\Large 0} &
\begin{matrix} 1&0\\0&-1 \end{matrix}
\end{pmatrix} \]
```

$$\begin{pmatrix} \begin{matrix} 1&0\\0&1 \end{matrix} & \text{\Large 0} \\ \text{\Large 0} & \begin{matrix} 1&0\\0&-1 \end{matrix} \end{pmatrix}$$

4-2-39

在行内公式中，有时需要使用很小的矩阵，这可以由 amsmath 提供的 smallmatrix 环境得到。smallmatrix 环境不给矩阵加括号，需要手工在外面添加括号（参见 4.3.4 节）：

```
复数 $z = (x,y)$ 也可用矩阵 \begin{math}
\left( \begin{smallmatrix}
x & -y \\ y & x
\end{smallmatrix} \right)
\end{math} 来表示。
```

复数 $z = (x,y)$ 也可用矩阵 $\left(\begin{smallmatrix} x & -y \\ y & x \end{smallmatrix} \right)$ 来表示。

4-2-40

在上下标特别是求和式的上下限中，有时需要使用好几行的内容，此时可以用 amsmath 提供的 \substack 命令排版，效果相当于只有一列的无括号的矩阵：

```
\[
\sum_{\substack{0<i<n \\ 0<j<i}} A_{ij}
\]
```

$$\sum_{\substack{0<i<n \\ 0<j<i}} A_{ij}$$

4-2-41

① 一些数学字体包，如 yhmath[99]，也提供了类似的反向省略号，不过 yhmath 的 \adots 的实现有问题，在不同字号下会变形。

类似地，subarray 环境也用在这种情况，但可以指定对齐方式为 l（左对齐）、c（居中）或 r（右对齐）：

4-2-42
```
\[  \sum_{\begin{subarray}{l}
  i<10 \\ j<100 \\ k<1000
\end{subarray}} X(i,j,k)  \]
```

$$\sum_{\substack{i<10\\ j<100\\ k<1000}} X(i,j,k)$$

matrix 等矩阵环境默认至多只有 10 列，直接使用多于 10 列的矩阵会产生错误。这个最大列数的限制由计数器 MaxMatrixCols 控制的，可以通过 \setcounter 等计数器命令临时或全局调整，如：

4-2-43
```
\[  \setcounter{MaxMatrixCols}{15}
\begin{Bmatrix}
0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 1 & 1 & 0 & 1\\
1 & 1 & 1 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 0
\end{Bmatrix}  \]
```

$$\begin{Bmatrix} 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 1 & 1 & 0 & 1\\ 1 & 1 & 1 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 \end{Bmatrix}$$

amsmath 提供的矩阵中，每列元素都是居中对齐的，mathtools 宏包进一步提供了带星号的 matrix*, pmatrix* 等矩阵环境，可以指定可选的列对齐方式。例如，数字长短不一的整数矩阵，往往右对齐显得更为清晰：

4-2-44
```
% \usepackage{mathtools}
\[  \begin{pmatrix*}[r]
  10 & -10 \\ -20 &  3
\end{pmatrix*}  \]
```

$$\begin{pmatrix*}[r] 10 & -10\\ -20 & 3 \end{pmatrix*}$$

一个比较原始的 LaTeX 命令 \bordermatrix 可以输入左、上边缘有边注的矩阵，它的语法和原始的 \matrix, \pmatrix 一样，与一般 LaTeX 不同：

4-2-45
```
\[  \bordermatrix{
  & 1 & 2 & 3 \cr
1 & A & B & C \cr
2 & D & E & F \cr}  \]
```

$$\bordermatrix{ & 1 & 2 & 3 \cr 1 & A & B & C \cr 2 & D & E & F \cr}$$

此外，Voß [281] 提供了 \bordermatrix 的推广形式，可以改变边注位置和括号。

矩阵可能需要在内部画线或画虚线表示分块，或者在矩阵外添加更复杂的括号和标注，或者使用复杂的对齐方式（如按小数点对齐）和可变化的距离，这时就必须回到基本 LaTeX 的 array 环境，手工调整相关的细节，我们将在 5.1 节（第 285 页）表格排版的部分继续这一话题。

4.3　符号与类型

如果说有限的几种数学结构确定了数学公式的骨架，那么无比丰富的数学符号才真正构成数学公式的血肉。LaTeX 的数学符号一共有多少种？如果不算拉丁字母的各种字体形式，默认的 Computer Modern 字体中的数学符号加上 $\mathcal{AMS}$ 数学符号，大约一共有 400 多个。而如果算上其他各种数学字体宏包的符号，这个数字将更为庞大。“符号大全” Pakin [192] 所收录的近 6000 个符号，半数都是数学符号。有条理地归纳这些数学符号，熟记常用符号的命令，了解符号的分类和使用规律，才可能真正熟练地使用 LaTeX 的数学功能。

TeX 中的数学符号是由专门的数学字体提供的。有的符号可以直接从键盘上输入，如字母 a, b, x 或符号 $+, =, ()$，但大部分的符号需要使用 LaTeX 命令输入。按照符号的意义和排版方式的不同，数学符号可以分成 8 个类别[126, Chapter 17]：普通符号、巨算符、二元运算符、关系符、开符号、闭符号、标点和变量族。变量族（一般就是字母）和普通符号性质类似，二元运算符与关系符性质类似，开符号和闭符号合起来就是完整的括号，因此下面把它们归并为同类，分类说明每类数学符号。

4.3.1　字母表与普通符号

字母表是数学符号最基本的内容。拉丁字母 A, B, C, x, y, z 等都可以直接从键盘输入，默认使用意大利形状。

数学字母可以使用多种字体，数字字体命令与正文字体命令语法类似，如 $\mathbf{X}$ 得到直立粗体的 $\mathbf{X}$。标准 LaTeX 提供的默认数学字母的字体见表 4.2（本书未使用默认字体，默认的 CM/LM 字体的效果与这里有差别）。

数学公式中，只有变量使用默认的意大利体，数学常数 e 通常使用罗马体的 e。类似地，虚数单位 i 也应该使用直立的罗马体 i。可以通过自定义命令简化某些常用常量的输入，如：

```
\newcommand\mi{\mathrm{i}}
\newcommand\me{\mathrm{e}}
```

4-3-1

表 4.2 标准 LaTeX 默认提供的数学字母字体

类别	字体命令	输出效果
数学环境的默认字体	\mathnormal	*ABCHIJXYZabchijxyz*12345
意大利体	\mathit	*ABCHIJXYZabchijxyz12345*
罗马体	\mathrm	ABCHIJXYZabchijxyz12345
粗体	\mathbf	**ABCHIJXYZabchijxyz12345**
无衬线体	\mathsf	ABCHIJXYZabchijxyz12345
打字机体	\mathtt	ABCHIJXYZabchijxyz12345
手写体（花体）†	\mathcal	$\mathcal{ABCHIJXYZ}$

† LaTeX 默认字体只支持大写字母，使用一些专业字体包可能支持小写字母：$\mathcal{ABCHIJXYZabchijxyz}$。

默认的数学字母字体是由 \mathnormal 产生的，它使用与正文不同的数学字体，字母之间的间距比正文要大一些，而 \mathit, \mathrm, \mathbf, \mathsf, \mathtt 几个数字字体的命令通常则直接使用与正文文字所对应的相同的字体，字母之间的间距与正文中基本一致（但没有连字）。因此，使用 xyz 通常表示三个字母的乘积 xyz（距离较大），而如果要表示一个多字母的长变量名 xyz，则应该使用 xyz 等形式。

表 4.3 常见的数学字母字体包

类别	字体命令	输出效果	宏包及说明
黑板粗体	\mathbb	$\mathbb{ABCXYZ}$	amssymb，仅大写字母 †
	\mathbb	$\mathbb{ABCXYZabcxyz123890}$	bbold
	\mathbbm	$\mathbbm{ABCXYZabcxyz12}$	bbm，数字仅有 1 和 2
花体	\mathscr	$\mathscr{ABCXYZ}$	mathrsfs，仅大写字母
	\mathcal	$\mathcal{ABCXYZ}$	euscript 加 eucal 选项，仅大写字母
哥特体	\mathfrak	$\mathfrak{ABCXYZabcxyz123890}$	amssymb 或 eufrak

† 此外 $\Bbbk$ 可以得到 $\Bbbk$。

利用数学字体包，还可以得到其他一些数学字母字体，常见的字体包见表 4.3。其中 amssymb[8] 是最常用的额外数学字体包，它提供了几种常用数学字母字体的访问①和大量新的数学符号。amssymb 扩展标准 LaTeX 的这部分符号通常称为 $\mathcal{AMS}$ 符号（后面

① 这个功能可以由 amssymb 的子宏包 amsfonts 单独完成，不过通常直接使用 amssymb 就可以了。

几节会陆续见到），除了 amssymb，许多其他改变整体字体风格的数学字体包也提供了适合其风格的 $\mathcal{AMS}$ 符号。类别相同而风格不同的数字字体包有很多，有的仅支持一两种特殊字体，如提供花体字母的 mathrsfs、euscript[170]，提供黑板粗体的 bbm[104]、bbold[115] 和 dsfont[135]，提供哥特体字母的 eufrak[171]。也有的会改变整体的数学字体甚至正文风格，如 txfonts[220]、pxfonts[219]、mathdesign[196] 和 fourier[30] 等。有关数学字体的比较可参见 Hartke [100]。

除了拉丁字母，数学公式中还经常使用希腊字母和少量其他类型的字母，见表 4.4、表 4.5 和表 4.6。

表 4.4　小写希腊字母

α	\alpha	β	\beta	γ	\gamma	δ	\delta
ϵ	\epsilon	ζ	\zeta	η	\eta	θ	\theta
ι	\iota	κ	\kappa	λ	\lambda	μ	\mu
ν	\nu	ξ	\xi	π	\pi	ρ	\rho
σ	\sigma	τ	\tau	υ	\upsilon	ϕ	\phi
χ	\chi	ψ	\psi	ω	\omega		
ε	\varepsilon	ϑ	\vartheta	$\varkappa$	\varkappa†	ϖ	\varpi
ϱ	\varrho	ς	\varsigma	φ	\varphi	F	\digamma†

前面带 var 的命令是原来字母的变体；\digamma 是 \gamma 的变体。

†$\mathcal{AMS}$ 符号，需要 amssymb 或类似的宏包。

表 4.5　大写希腊字母

Γ	\Gamma	Δ	\Delta	Θ	\Theta	Λ	\Lambda
Ξ	\Xi	Π	\Pi	Σ	\Sigma	Υ	\Upsilon
Φ	\Phi	Ψ	\Psi	Ω	\Omega		
$\varGamma$	\varGamma	$\varDelta$	\varDelta	$\varTheta$	\varTheta	$\varLambda$	\varLambda
$\varXi$	\varXi	$\varPi$	\varPi	$\varSigma$	\varSigma	$\varUpsilon$	\varUpsilon
$\varPhi$	\varPhi	$\varPsi$	\varPsi	$\varOmega$	\varOmega		

一些大写希腊字母与拉丁字母形状相同，如 A, B，在数学公式中不使用，因而没有对应的命令。
大写希腊字母一般用正体；前面带 var 的命令是原来字母的变体（倾斜）形式，需要 amsmath 宏包支持。
也可以改用 \mathnormal 字体得到倾斜形式的大写希腊字母。

表 4.6　希伯来字母

ℵ	\aleph	ℶ	\beth†	ℸ	\daleth†	ℷ	\gimel†

†$\mathcal{AMS}$ 符号，需要 amssymb 或类似的宏包。

按 ISO 标准对物理和其他科技文档的要求[114、261]，数学常数 π 通常使用直立体（斜体的 π 则作为变量），这可以使用数学字体宏包 upgreek[228] 得到直立体的希腊字母以配合默认字体使用，其命令名是在标准的希腊字母命令名前加 up，如 \uppi（参见表 4.7）。许多其他数学字体包（如 txfonts, fourier, mathdesign）也提供类似的直立体希腊字母，其命令各有不同，可参见对应的字体宏包文档。

表 4.7 直立体希腊字母（小写）。需要 upgreek 或类似字体包

$\upalpha$	\upalpha	$\upbeta$	\upbeta	$\upgamma$	\upgamma	$\updelta$	\updelta
$\upepsilon$	\upepsilon	$\upzeta$	\upzeta	$\upeta$	\upeta	$\uptheta$	\uptheta
$\upiota$	\upiota	$\upkappa$	\upkappa	$\uplambda$	\uplambda	$\upmu$	\upmu
$\upnu$	\upnu	$\upxi$	\upxi	$\uppi$	\uppi	$\uprho$	\uprho
$\upsigma$	\upsigma	$\uptau$	\uptau	$\upupsilon$	\upupsilon	$\upphi$	\upphi
$\upchi$	\upchi	$\uppsi$	\uppsi	$\upomega$	\upomega		
$\upvarepsilon$	\upvarepsilon	$\upvartheta$	\upvartheta	$\upvarpi$	\upvarpi	$\upvarrho$	\upvarrho
$\upvarsigma$	\upvarsigma	$\upvarphi$	\upvarphi				

与正文中的字母一样，数学字母也可以加上重音符号，见表 4.8。数学重音（math accents）与正文中的重音使用完全不同的命令，如 \$\hat a\$ 得到 $\hat{a}$。其中有些重音是可延长的宽重音符号，与 \overline 等命令作用类似，如 \$\widehat{abc}\$ 得到 $\widehat{abc}$。后面以 \sp 开头的命令则是上标形式的重音（需要 amsxtra[66]），如 \$(a+b)\sphat\$ 得到 $(a+b)^{\frown}$，一般用于给很长的一段公式加标记。数学重音也可以叠加使用，如 \$\hat{\bar a}\$ 可得到 $\hat{\bar{a}}$。

表 4.8 数学重音

$\acute{a}$	\acute	$\grave{a}$	\grave	$\ddot{a}$	\ddot	$\tilde{a}$	\tilde
$\bar{a}$	\bar	$\breve{a}$	\breve	$\check{a}$	\check	$\hat{a}$	\hat
$\vec{a}$	\vec	$\dot{a}$	\dot	$\mathring{a}$	\mathring		
$\widetilde{abc}$	\widetilde	$\widehat{abc}$	\widehat				
$\ddot{a}$	\dddot†	$\ddot{\ddot{a}}$	\ddddot†				
$(abc)^{\smile}$	\spbreve	$(abc)^{\vee}$	\spcheck	$(abc)^{\cdot}$	\spdot	$(abc)^{\cdot\cdot}$	\spddot
$(abc)^{\cdots}$	\spdddot	$(abc)^{\frown}$	\sphat	$(abc)^{\sim}$	\sptilde		

†需要 amsmath 宏包。

\sp 开头的命令需要 amsxtra 宏包。

一些数学字体包还会提供更多的重音，例如 yhmath[99] 提供了（适合 Computer

Modern 字体的）\widetriangle、\wideparen 等命令[①]：

$\widetriangle{abc}$	\widetriangle	$\wideparen{abc}$	\wideparen

其中的 \wideparen 较为常用，也在其他一些数学字体包，如 MnSymbol[29]、mathdesign[196]、fourier[30] 等，也提供了适合对应字体的 \wideparen 版本。

如果还需要更为特殊的重音符号，可以使用 accents[26] 宏包来生成“伪重音”。accents 宏包重定义了标准 LaTeX 的数学重音机制，同时提供了更一般的命令：\accentset{⟨符号⟩}{⟨内容⟩} 可以在内容上方增加任意符号作为重音，\underaccent{⟨重音命令⟩}{⟨内容⟩} 可以产生在内容下方的特殊重音，而 \undertilde 则是宽重音符号 \widetilde 的下方版本：

```
% \usepackage{accents}
$\accentset{*}{A}$, $\accentset{@}{X}$,
$\underaccent{\check}{a}$,
$\underaccent{\hat}{b}$,
$\undertilde{abc}$
```

$\overset{*}{A}, \overset{@}{X}, \underset{\check{}}{a}, \underset{\hat{}}{b}, \underset{\sim}{abc}$

4-3-2

不过这个宏包有一些兼容性的问题：一是它必须放在 amsmath 的后面，二是如果使用了 fontspec（包括 xeCJK 和 ctex 文档类），则还要给 fontspec 加 [no-math] 选项禁止设置数学字体，才能正常使用。也可以在引入宏包后用 savesym 宏包取消 accents 对原有重音命令的重定义来解决冲突（参见 7.1.2.2 节）。

除了各种字母，LaTeX 还提供了许多数学环境的普通符号（ordinary symbols），见表 4.9。它们产生的数学间距与字母相同，与字母的唯一区别是没有数学字母那么多不同的字体[②]。普通符号通常是一些字母的变形、一元运算符或者是单纯的图形符号。

表 4.9 中，\imath（$\imath$）和 \jmath（$\jmath$）就是不加点的拉丁字母 i 和 j，与正文中一样，它们用在给 i 和 j 加重音的时候，即应该用 $\check{\imath}$ 而不是 $\check{i}$，用 $\vec{\jmath}$ 而不是 $\vec{j}$。符号 ℓ 就是小写拉丁字母 l，提供这个符号是为了避免与数字 1 混淆。符号 $\prime$ 正是撇号 ′ 的非上标形式。符号 $\surd$ 就是没有横线的根号。

此外，还有一些符号可同时用在文本模式和数学模式中（大多用在文本模式中），可以作为表 2.3 的补充，见表 4.10。

① 由于历史原因，yhmath 提供的个别命令有一些缺陷和错误，如结合了 \mathring 与 \wideparen 功能的 \mathring，不如直接嵌套 \mathring 与 \wideparen 效果好，又如反向省略号 \adots 的问题（参见 4.2.5 节），再如其 amatrix 环境的定义会导致错误，因此使用时应该小心。

② 事实上，数字字母一般只包括拉丁字母，希腊字母和希伯来字母在 LaTeX 中也是字体固定的普通符号。

表 4.9 数学普通符号

$\hbar$	\hbar	$\imath$	\imath	$\jmath$	\jmath	ℓ	\ell
$\wp$	\wp	$\Re$	\Re	$\Im$	\Im	∂	\partial
∞	\infty	$\prime$	\prime	$\emptyset$	\emptyset	∇	\nabla
$\surd$	\surd	$\top$	\top	$\bot$	\bot	$\angle$	\angle
$\triangle$	\triangle	$\forall$	\forall	$\exists$	\exists	$\neg$	\neg
$\flat$	\flat	$\natural$	\natural	$\sharp$	\sharp	$\clubsuit$	\clubsuit
$\diamondsuit$	\diamondsuit	$\heartsuit$	\heartsuit	$\spadesuit$	\spadesuit	$\backslash$	\backslash†
$\backprime$	\backprime	$\hslash$	\hslash	$\varnothing$	\varnothing	$\vartriangle$	\vartriangle
$\blacktriangle$	\blacktriangle	$\triangledown$	\triangledown	$\blacktriangledown$	\blacktriangledown	$\square$	\square
$\blacksquare$	\blacksquare	$\lozenge$	\lozenge	$\blacklozenge$	\blacklozenge	Ⓢ	\circledS
$\bigstar$	\bigstar	$\sphericalangle$	\sphericalangle	$\measuredangle$	\measuredangle	$\nexists$	\nexists
$\complement$	\complement	$\mho$	\mho	$\eth$	\eth	$\Finv$	\Finv
$\diagup$	\diagup	$\Game$	\Game	$\diagdown$	\diagdown	$\Bbbk$	\Bbbk

从 \backprime 开始是 $\mathcal{AMS}$ 符号。

†\backslash 同时也是长度可变的定界符，并有一个同形的二元运算符 \setminus。

表 4.10 可同时用在文本和数学模式中的符号

#	\#	&	\&	%	\%	$	\$
_	_	{	\{	}	\}		
¶	\P	§	\S	†	\dag	‡	\ddag
©	\copyright	£	\pounds	…	\ldots†		
✓	\checkmark	®	\circledR	✠	\maltese	¥	\yen

最后一行是 $\mathcal{AMS}$ 符号。

{ 和 } 是定界符（括号），参见 4.3.4 节。

†数学省略号参见 4.3.5 节。

现在我们回到数学字体，很多时候我们需要粗体的数学符号（bold math，而不是与正文字体一样的 \mathbf），例如数学中常用粗斜体的字母 $\boldsymbol{a}$ 表示向量，又如我们在章节标题中需要用粗体的“**勾股定理 $a^2+b^2=c^2$**”。传统上，LaTeX 使用 \boldmath 选择粗体数学公式，但要用在数学公式的外面，为了避免影响到后面的内容，通常还要把 \boldmath 与整个公式用单独的分组里面，如用 {\boldmath$a^2$} 输出 $\boldsymbol{a^2}$。

如果要在同一个公式中使用不同粗细的数学符号，\boldmath 的限制将是一场恶梦。amsmath 提供了单参数 \boldsymbol 命令，可以在数学公式中使符号变粗，如用 $\boldsymbol{v} = (0,1)$ 得到 $\boldsymbol{v} = (0,1)$。对于没有粗体形式的符号，用 \pmb 命令可以得到伪粗体的符号[①]，如用 $\pmb\sum$ 可得到 $\pmb\sum$。

不过，有时 \boldmath 和 \boldsymbol 并不能正确地找到许多数学符号的粗体，更好的办法是使用 bm 宏包[48] 来选择数学粗体。bm 宏包提供两个简单的单参数命令 \bm 和 \hm，其中 \bm 选择粗体数学符号，\hm 选择更粗的加重体数学符号（heavy math，只有个别数学字体包支持加重体的数学符号）。如果符号的粗体形式不存在，bm 宏包一般会自动选择伪粗体来表示。例如：

```
% \usepackage{bm}
% \hm 的效果需要实际字体支持
\textbf{勾股定理 $\bm{a^2+b^2=c^2}$}
\[  \bm u + \bm v = (1,0) + (0,1)  \]
\[  \hm\int > \bm\int > \int  \]
```

勾股定理 $a^2+b^2=c^2$

$$\boldsymbol{u} + \boldsymbol{v} = (1,0) + (0,1)$$

$$\boldsymbol{\int} > \int > \int$$

4-3-3

为了避免疏忽造成的错误，在 LaTeX 中需要使用粗体数学符号时，通常最好总是使用 bm 宏包的 \bm 完成。少数 bm 找不到对应粗体而又没有使用伪粗体的时候，可以直接使用 amsmath 伪粗体的 \pmb 命令。

数学符号的类型是在数学符号定义时就预先确定的[33、142]，符号的类型又直接决定了不同符号的排版间距。例如 $a+b$ 中加号 + 是二元运算符，因而在加号前后就会产生比普通符号更多的间距。可以通过把数学符号或公式的一部分放在单独的分组里面，把这部分公式看做一个单独的普通符号，使用 \mathord 命令也有同样的效果（通常是多余的），可以用这种方式精细地控制数学间距，如：

① 伪粗体是把符号稍稍错位地连续输出多次得到的，旧方式下 CJK 宏包处理字体贫乏的汉字粗体也常用这种方式。

```
\begin{gather*}
a+b \quad a{+}b \quad a\mathord{+}b \\
\max n \quad {\max} n
  \quad \mathord{\max} n
\end{gather*}
```

$$a+b \quad a{+}b \quad a\mathord{+}b$$
$$\max n \quad {\max} n \quad \mathord{\max} n$$

练 习

4.2 排版公式

$$e^{\pi i} + 1 = 0 \tag{4.3}$$

4.3 正确排版这段话：

> 空集 $\varnothing$ 的基数是 0，自然数集 $\mathbb{N} = \{1, 2, 3, \ldots\}$ 的基数是 $\aleph_0$，则实数集 $\mathbb{R}$ 的基数 $\#\mathbb{R} = \aleph_1 = 2^{\aleph_0}$.

4.4 一个矩阵的转置应该如何表示？是 M'、$\mathbf{M}^T$ 还是 $\boldsymbol{M}^{\mathrm{T}}$，抑或是别的什么记法？说说你的理由。

4.5 $\Re$ 和 $\Im$ 是什么符号？`\partial` 和 `\infty` 是什么意思？试着理解符号表中所有的命令名称，合上书，也不借助编辑器中的符号面板按钮，看看你能否用键盘打出你想要的符号。

4.3.2 数学算子

下面来看数学算子（math operator）。数学算子通常会与前后的符号都有一小段间距，例如函数 $\sin x$ 中的 sin 就与变量 x 分开，$\max\limits_{x}\min\limits_{y} f(x,y)$ 的两部分也不会挤在一起。

数学算子分为两种，第一类是类似求和号 $\sum$、积分号 $\int$ 的算子，它们的大小是随显示公式和行内公式变化的，而且通常比一般的数学符号大一些，因而又被称为巨算符（large operator）。

除了表 4.11 中的积分号 `\int`，LaTeX 还提供了一个不能改变大小的小积分号 `\smallint` $\smallint$。

使用其他数学字体包可能会提供更多的巨算符，比如说二维的环路积分 $\oiint$ 或定向积分 $\oint$，可参见符号表 Pakin [192]，或查看实际文档中使用的数学字体包文档。

表 4.11　大小可变的运算符（巨算符）

$\sum\ \Sigma$	\sum	$\prod\ \Pi$	\prod	$\coprod\ \amalg$	\coprod
$\int\ \int$	\int	$\oint\ \oint$	\oint		
$\bigcup\ \cup$	\bigcup	$\biguplus\ \uplus$	\biguplus	$\bigsqcup\ \sqcup$	\bigsqcup
$\bigvee\ \vee$	\bigvee	$\bigwedge\ \wedge$	\bigwedge	$\bigcap\ \cap$	\bigcap
$\bigodot\ \odot$	\bigodot	$\bigoplus\ \oplus$	\bigoplus	$\bigotimes\ \otimes$	\bigotimes
$\iint\ \iint$	\iint	$\iiint\ \iiint$	\iiint	$\iiiint\ \iiiint$	\iiiint
$\int\cdots\int\ \int\cdots\int$	\idotsint				

最后两行的积分符号需要 amsmath。

注意不要把巨算符和形状相似的其他类型符号混淆，比如大写希腊字母 Σ（Sigma）与求和号 ∑（\sum），二元逻辑或运算符 ∨（\vee）与逻辑或巨算符 ⋁（\bigvee），它们来自数学字体的不同符号，意义、符号大小、产生的间距、上下标的排版方式等都有区别。

数学算子通常都可以带上下标，正如 4.1 节所说，如果不使用 amsmath 的选项修改，在显示公式中，各种积分号的上下标默认都在角标位置，而其他巨算符则在上下方（作为上下限），例如：

```
\[
\mathcal{F}(x) = \sum_{k=0}^\infty
  \oint_0^1 f_k(x,t) \,\mathrm{d}t
\]
```

$$\mathcal{F}(x) = \sum_{k=0}^{\infty} \oint_0^1 f_k(x,t)\,\mathrm{d}t$$

4-3-4

当然，也可以用 \limits 和 \nolimits 手工控制巨算子的位置为上下限位置还是角标位置，如例 4-3-6 和第 227 页的例 4-2-8。

```
但是，在行内公式中，不要使用形如
$\bigoplus\limits_{j=1}^n P_j$
的上下限，以免上下限与后面的文字
挤在一起，或是造成难看的不均匀行距。
```

但是，在行内公式中，不要使用形如 $\bigoplus\limits_{j=1}^{n} P_j$ 的上下限，以免上下限与后面的文字挤在一起，或是造成难看的不均匀行距。

注意积分式的写法，积分式中的微元 dt 里面，微分算子 d 应该使用直立罗马体，后面的变量则仍是默认的意大利体，并且用 \, 与前面的被积函数分开：

4-3-5

```
\[  \int f(x) \,\mathrm{d} x  \]
```

$$\int f(x)\,\mathrm{d}x$$

虽然在数学上微分算子 d 也是一个数学算子，但因为它太短了，在排版时并不像其他数学算子一样要在 d 与变元之间添加间距。不过微元与被积函数、多个微分变元之间，仍然需要留有间距：

4-3-6

```
\newcommand\diff{\,\mathrm{d}}
\[  \iiint\limits_{0<x,y,z<1} f(x,y,z)
  \diff x \diff y \diff z  \]
```

$$\iiint\limits_{0<x,y,z<1} f(x,y,z)\,\mathrm{d}x\,\mathrm{d}y\,\mathrm{d}z$$

事实上，这类只有一个字符的数学算子，一般都只作为一个普通数学符号排版，我们在 4.3.1 节见到的 Laplace 算子 Δ（大写希腊字母 \Delta，也可以用 △ \triangle）、偏微分算子 ∂（\partial）、梯度算子 ∇（\nabla）等都是主要作为没有额外间距的单字符算子使用的。

第二类数学算子是文字名称的算子，它们用直立的罗马体排印，如 $\log x$, $\lim f(t)$ 中的 log 和 lim。前者是不带上下限的“纯”算子，一般就是常用的数学函数（见表 4.12）；后者是可以带有上下限的数学算子，用法与求和、积分号类似（见表 4.13）。

表 4.12 不带上下限的数学算子名

log	\log	lg	\lg	ln	\ln	sin	\sin	arcsin	\arcsin
cos	\cos	arccos	\arccos	tan	\tan	arctan	\arctan	cot	\cot
sinh	\sinh	cosh	\cosh	tanh	\tanh	coth	\coth	sec	\sec
csc	\csc	arg	\arg	ker	\ker	dim	\dim	hom	\hom
exp	\exp	deg	\deg						

不带上下限的函数式算子，其上标是角标形式，后面的数学参数（并非 LaTeX 宏参数）如果不止一项，应该带上括号：

4-3-7

```
$  \cos 2x = \cos(x+x)
= \cos^2 x - \sin^2 x  $
```

$$\cos 2x = \cos(x+x) = \cos^2 x - \sin^2 x$$

表 4.13　带上下限的数学算子名

lim	\lim	lim sup	\limsup	lim inf	\liminf	max	\max
min	\min	sup	\sup	inf	\inf	det	\det
Pr	\Pr	gcd	\gcd				
$\underline{\lim}$	\varliminf	$\overline{\lim}$	\varlimsup	inj lim	\injlim	proj lim	\projlim
$\varinjlim$	\varinjlim	$\varprojlim$	\varprojlim				

后两行需要 amsmath 宏包。

带上下限的数学算子使用起来与巨算符相似[①]：

```
\begin{equation}
  \varlimsup_{k\to\infty} A_k = \lim_{J\to\infty} \lim_{K\to\infty}
    \bigcap_{j=1}^J \bigcup_{k=j}^K A_k
\end{equation}
```

4-3-8

$$\varlimsup_{k\to\infty} A_k = \lim_{J\to\infty} \lim_{K\to\infty} \bigcap_{j=1}^{J} \bigcup_{k=j}^{K} A_k \tag{4.4}$$

尽管 LaTeX 已经预定义了许多算子名，实际中仍不免捉襟见肘。因而 amsmath 提供了 \DeclareMathOperator 命令来声明新的算子名，其用法与 \newcommand 相似。如果使用带星号的 \DeclareMathOperator* 命令，则可以声明带上下限的数学算子。这两个命令都只用于导言区：

```
% 导言区 \usepackage{amsmath}
\DeclareMathOperator{\card}{card}          % 集合基数
\DeclareMathOperator*{\esssup}{ess\,sup}   % 本性上确界
```

4-3-9

如果只是在一两个公式中临时使用，也可以不专门定义命令，而是使用 \operatorname{(内容)} 来表示。带星的 \operatorname* 则是上下限形式的上下标。例如：

```
\[  \operatorname*{Prob}_{\{1,\ldots,n\}}
  (\bar X) =
\operatorname{card}(\varnothing)/n = 0. \]
```

$$\operatorname*{Prob}_{\{1,\ldots,n\}}(\bar{X}) = \operatorname{card}(\varnothing)/n = 0.$$

4-3-10

① 这里 \to 就是右箭头符号 → 的命令，参见 4.3.3 节。

可以使用 \mathop 命令来让 TEX 将其参数看做是数学算子。例如，也可以使用 $\mathop{\mathrm{sin}} x$ 来代替 $\sin x$ 得到 $\sin x$，实际上 \operatorname 等命令的定义就离不开 \mathop。下面这个例子可以解释如何综合使用 \mathord 和 \mathop 连续改变符号的类型，以实现 \sideset 的功能：

```
% 用原始命令实现 \sideset 的功能
\[  \mathop{\mathord{\sum}'}
\limits_{i=1}^n A_n  \]
```

$$\sum_{i=1}^{n}{}' A_n$$

不过在可能的时候，应该还是使用 \operatorname、\sideset 这类高级的命令。

我们现在来改进部分单字符数学算子的排版。前面我们在排版积分微元 $\mathrm{d}x$ 时，经常需要在前面手工加上 \，表示间距，但如果要排版 $\frac{\mathrm{d}x}{\mathrm{d}t}$ 这种分式，又不应该加上间距了。如果要给微分算子 d 定义一个命令 \dif，能否让它在不同的情况下都产生正确的间距呢？下面的定义可以解决这个问题：

4-3-11

```
% 导言区 \DeclareMathOperator\dif{d\!}
\[
\int_0^1 \int_0^1 f(x,y) \int_0^1 \frac{\dif z}{g(x,y,z)} \dif x\dif y
\]
```

$$\int_0^1 \int_0^1 f(x,y) \int_0^1 \frac{\mathrm{d}z}{g(x,y,z)} \mathrm{d}x\,\mathrm{d}y$$

这个定义来自 commath 宏包[197]，它利用 \DeclareMathOperator 定义算子保证微分算子前面的间距，同时用 \! 去掉了后面的间距，以符合习惯。

此外，关于取模和同余的符号 mod，LaTeX 与 amsmath 还专门定义一组命令，它们不是数学算子，但效果和用法与数学算子类似。基本 LaTeX 提供了取模的二元运算符 \bmod 和单参数命令 \pmod，amsmath 补充了两个单参数命令 \mod 和 \pod。除 \bmod 外，另外三个命令都会在前面增加一段较大的间距，它们的用法示例如下：

$r = m \bmod n$	`r = m \bmod n`
$x \equiv y \pmod b$	`x\equiv y \pmod b`
$x \equiv y \mod c$	`x\equiv y \mod c`
$x \equiv y \pod d$	`x\equiv y \pod d`

练 习

4.6 排版 Gauss-Bonnet 公式：

$$\oint_C \kappa_g \,\mathrm{d}s + \iint_D K \,\mathrm{d}\sigma = 2\pi - \sum_{i=1}^{n} \alpha_i. \tag{4.5}$$

4.7 排版离散分布随机变量的方差公式，注意概率、期望和方差几个数学算子：

$$\mathrm{Var}(X) = \mathrm{E}(X-\mu)^2 = \sum_{j=1}^{\infty} (x_j - \mu)^2 \Pr(X = x_j), \qquad \text{其中 } \mu = \mathrm{E}X. \tag{4.6}$$

4.8 排版 Fourier 积分公式：

$$\lim_{N\to+\infty} \frac{1}{2\pi} \int_{-N}^{N} \hat{f}(\lambda) \mathrm{e}^{\mathrm{i}\lambda x} \,\mathrm{d}\lambda = f(x). \tag{4.7}$$

4.3.3　二元运算符与关系符

二元运算符与关系符都是用在公式中间，在符号的两边留有一定间距。运算符的间距小一些，关系符的间距略大一些。尽管二者的差别非常小[①]，肉眼基本不能区分，不过，类型的差别可以帮助 TEX 区分公式的不同情况，好像 TEX 理解公式一样，例如等式 `$0-2=-2$` 即

$$0 - 2 = -2$$

左边的减号和右边的负号与数字 2 间距的就不一样，因为左边的减号两边都是普通符号，而右边的减号左边是一个关系符。当然，在大多数情况下，我们并不需要关心这些细节，LATEX 会自动处理好一切。

二元运算符和关系符还有一个共同点，就是当 TEX 需要在行内公式中断行时，可以（而且也只能）在它们后面断开。因此我们可以随意地写 $1+2+3+4+5+6+7+8+9+10+11+12+13+14+15+16+17+18+19+20+21+22+23+24+25+26+27+28+29+30=465$，而不必担心公式太长的问题。

能从键盘上直接输入的二元运算符有加号 +、减号 - 和星号 *。不过斜线形式的除号 / 并不是二元运算符，只是普通符号[②]，因为即使它表示除法也不额外增加间距。表示差集的符号 `\setminus` 与普通符号 `\backslash` 是同一个字符，但类型不同，因而会产生不同的间距，对比下面的例子：

① 确切地说，只有 1/18 个 em 宽。
② 同时也是可变长的定界符。

4-3-12

```
群 $G$ 的 $(H,K)$-双陪集为
$H\backslash G/K$。

$S\cup T = (S\cap T)\cup (S\setminus T)$
```

群 G 的 (H,K)-双陪集为
$H\backslash G/K$。
$S\cup T = (S\cap T)\cup (S\setminus T)$

表 4.14 LaTeX 中的二元运算符

◁	\triangleleft			▷	\triangleright		
△	\bigtriangleup			▽	\bigtriangledown		
∧	\wedge 或 \land	∨	\vee 或 \lor	∩	\cap	∪	\cup
‡	\ddagger	†	\dagger	⊓	\sqcap	⊔	\sqcup
⊎	\uplus	⨿	\amalg	⋄	\diamond	•	\bullet
≀	\wr	÷	\div	⊙	\odot	⊘	\oslash
⊗	\otimes	⊕	\oplus	∓	\mp	±	\pm
∘	\circ	◯	\bigcirc	\	\setminus	·	\cdot
*	\ast 或 *	×	\times	⋆	\star		

表 4.15 $\mathcal{AMS}$ 二元运算符

∔	\dotplus	⧹	\smallsetminus	⊺	\intercal
⋒	\Cap 或 \doublecap	⋓	\Cup 或 \doublecup		
⊼	\barwedge	⊻	\veebar	⩛	\doublebarwedge
⊟	\boxminus	⊡	\boxdot	⊞	\boxplus
⋉	\ltimes	⋊	\rtimes	⋇	\divideontimes
⋋	\leftthreetimes	⋌	\rightthreetimes		
⋏	\curlywedge	⋎	\curlyvee	∙	\centerdot
⊝	\circleddash	⊛	\circledast	⊚	\circledcirc
◁	\lhd	⊴	\unlhd		
▷	\rhd	⊵	\unrhd		

二元关系符是 LaTeX 中数量最为庞大的一类数学符号：除了有普通的二元关系符及它们的否定形式外，关系符中各种箭头通常也单独列为一类，标准字体加上 $\mathcal{AMS}$ 字符，加起来大约有 200 个符号。

从键盘上可以直接输入的二元关系符有等号 =，大于号 >，小于号 < 和表示集合的关系符号 :。这里应注意在数学环境中键盘上的冒号得到的是一个二元关系符，与其

两边的内容间距相等，这与间距不均匀的数学标点 \colon 不同（参见 4.3.5 节），例如：

```
\[  \{ x \in \mathbb{N} : f(x) = 0 \}  \]
```

$$\{x \in \mathbb{N} : f(x) = 0\}$$

4-3-13

一般地，二元关系符的否定可以在关系符前加 \not 命令得到，如 $s\not\in T$ 就得到 $s \notin T$。此外，LaTeX 也为很多二元关系符的否定形式单独定义了命令，单独定义的命令有时会使用单独的字体符号，比直接用 \not 得到的效果更好，如 $s\notin T$ 得到的 $s \notin T$，斜线的位置更合理一些。因此，在使用否定二元关系符时，应该尽量使用单独的符号命令；在没有单独的符号时，才使用 \not 组合符号。

表 4.16　二元关系符及其否定形式

$=$	=	$\neq$	\neq 或 \ne	$:$	:		
$<$	<	$\not<$	\nless†	$>$	>	$\not>$	\ngtr†
$\leq$	\leq 或 \le	$\not\leq$	\nleq†	$\geq$	\geq 或 \ge	$\not\geq$	\ngeq†
$\in$	\in	$\notin$	\notin	$\ni$	\ni 或 \owns		
$\ll$	\ll			$\gg$	\gg		
$\prec$	\prec	$\not\prec$	\nprec†	$\succ$	\succ	$\not\succ$	\nsucc†
$\preceq$	\preceq	$\not\preceq$	\npreceq†	$\succeq$	\succeq	$\not\succeq$	\nsucceq†
		$\precneqq$	\precneqq†			$\succneqq$	\succneqq
$\sim$	\sim	$\not\sim$	\nsim†	$\approx$	\approx		
$\simeq$	\simeq			$\cong$	\cong	$\not\cong$	\ncong†
$\equiv$	\equiv			$\doteq$	\doteq		
$\subset$	\subset			$\supset$	\supset		
$\subseteq$	\subseteq	$\not\subseteq$	\nsubseteq†	$\supseteq$	\supseteq	$\not\supseteq$	\nsupseteq†
$\subsetneq$	\subsetneq†	$\varsubsetneq$	\varsubsetneq†	$\supsetneq$	\supsetneq†	$\varsupsetneq$	\varsupsetneq†
$\smile$	\smile			$\frown$	\frown		
$\perp$	\perp			$\models$	\models		
$\mid$	\mid	$\nmid$	\nmid†	$\parallel$	\parallel	$\nparallel$	\nparallel†
$\vdash$	\vdash	$\nvdash$	\nvdash†	$\dashv$	\dashv		
$\propto$	\propto			$\asymp$	\asymp		
$\bowtie$	\bowtie	$\Join$	\Join†				

†AMS 符号。

表 4.17 $\mathcal{AMS}$ 二元关系符及其否定形式

$\leqq$ \leqq	$\nleqq$ \nleqq	$\geqq$ \geqq	$\ngeqq$ \ngeqq
$\lneqq$ \lneqq	$\lvertneqq$ \lvertneqq	$\gneqq$ \gneqq	$\gvertneqq$ \gvertneqq
$\leqslant$ \leqslant	$\nleqslant$ \nleqslant	$\geqslant$ \geqslant	$\ngeqslant$ \ngeqslant
	$\lneq$ \lneq		$\gneq$ \gneq
$\lesssim$ \lesssim	$\lnsim$ \lnsim	$\gtrsim$ \gtrsim	$\gnsim$ \gnsim
$\lessapprox$ \lessapprox	$\lnapprox$ \lnapprox	$\gtrapprox$ \gtrapprox	$\gnapprox$ \gnapprox
$\precsim$ \precsim	$\precnsim$ \precnsim	$\succsim$ \succsim	$\succnsim$ \succnsim
$\precapprox$ \precapprox	$\precnapprox$ \precnapprox	$\succapprox$ \succapprox	$\succnapprox$ \succnapprox
$\subseteqq$ \subseteqq	$\nsubseteqq$ \nsubseteqq	$\supseteqq$ \supseteqq	$\nsupseteqq$ \nsupseteqq
$\subsetneqq$ \subsetneqq	$\varsubsetneqq$ \varsubsetneqq	$\supsetneqq$ \supsetneqq	$\varsupsetneqq$ \varsupsetneqq
$\vartriangleleft$ \vartriangleleft	$\ntriangleleft$ \ntriangleleft	$\vartriangleright$ \vartriangleright	$\ntriangleright$ \ntriangleright
$\trianglelefteq$ \trianglelefteq	$\ntrianglelefteq$ \ntrianglelefteq	$\trianglerighteq$ \trianglerighteq	$\ntrianglerighteq$ \ntrianglerighteq
$\shortmid$ \shortmid	$\nshortmid$ \nshortmid	$\shortparallel$ \shortparallel	$\nshortparallel$ \nshortparallel
$\vDash$ \vDash	$\nvDash$ \nvDash	$\Vdash$ \Vdash	$\nVdash$ \nVdash
$\Vvdash$ \Vvdash			$\nVDash$ \nVDash

表 4.18 没有否定形式的 $\mathcal{AMS}$ 二元关系符

$\eqslantless$ \eqslantless	$\eqslantgtr$ \eqslantgtr	$\approxeq$ \approxeq	
$\lessdot$ \lessdot	$\gtrdot$ \gtrdot	$\lll$ \lll	$\ggg$ \ggg
$\lessgtr$ \lessgtr	$\gtrless$ \gtrless	$\lesseqgtr$ \lesseqgtr	$\gtreqless$ \gtreqless
$\lesseqgtr$ \lesseqgtr	$\gtreqless$ \gtreqless	$\doteqdot$ \doteqdot	$\triangleq$ \triangleq
$\eqcirc$ \eqcirc	$\circeq$ \circeq	$\risingdotseq$ \risingdotseq	$\fallingdotseq$ \fallingdotseq
$\backsim$ \backsim	$\thicksim$ \thicksim	$\backsimeq$ \backsimeq	$\thickapprox$ \thickapprox
$\preccurlyeq$ \preccurlyeq	$\succcurlyeq$ \succcurlyeq	$\sqsubseteq$ \sqsubseteq	$\sqsupseteq$ \sqsupseteq
$\sqsubset$ \sqsubset	$\sqsupset$ \sqsupset	$\Subset$ \Subset	$\Supset$ \Supset
$\smallsmile$ \smallsmile	$\smallfrown$ \smallfrown	$\bumpeq$ \bumpeq	$\Bumpeq$ \Bumpeq
$\between$ \between	$\pitchfork$ \pitchfork	$\varpropto$ \varpropto	$\backepsilon$ \backepsilon
$\blacktriangleleft$ \blacktriangleleft	$\blacktriangleright$ \blacktriangleright	$\therefore$ \therefore	$\because$ \because

表 4.19　LaTeX 箭头符号

←	\leftarrow 或 \gets	↚	\nleftarrow†
→	\rightarrow 或 \to	↛	\nrightarrow†
⇐	\Leftarrow	⇍	\nLeftarrow†
⇒	\Rightarrow	⇏	\nRightarrow
↔	\leftrightarrow	↮	\nleftrightarrow†
⇔	\Leftrightarrow	⇎	\nLeftrightarrow†
⟵	\longleftarrow	⟶	\longrightarrow
⟸	\Longleftarrow	⟹	\Longrightarrow
⟷	\longleftrightarrow	⟺	\Longleftrightarrow
↦	\mapsto	⟼	\longmapsto
↩	\hookleftarrow	↪	\hookrightarrow
↼	\leftharpoonup	⇀	\rightharpoonup
↽	\leftharpoondown	⇁	\rightharpoondown
⇌	\rightleftharpoons		
↗	\nearrow	↘	\searrow
↙	\swarrow	↖	\nwarrow
↑	\uparrow	⇑	\Uparrow
↓	\downarrow	⇓	\Downarrow
↕	\updownarrow	⇕	\Updownarrow

最后三行垂直的箭头同时也是可延长的定界符。

†$\mathcal{AMS}$ 否定箭头。

有时需要在关系符上方添加额外的说明字符，比如表示定义的等号，除了可以用 $\mathcal{AMS}$ 符号 ≜，也可以在等号上面添加一个字母 d，即 $\stackrel{\text{d}}{=}$。可以用 \stackrel 命令得到这种在关系符上堆叠符号的效果，如：

```
\newcommand\defeq{\stackrel{\text{d}}{=}}
$f(x) \defeq ax^2+bx+c$
```

$$f(x) \stackrel{\text{d}}{=} ax^2+bx+c$$

4-3-14

不过，当添加的说明太长时，普通的等号、箭头就不够用了。为此，amsmath 提供了可延长的箭头命令 \xleftarrow 和 \xrightarrow，可以在其上下方添加说明，命令参数是上方的说明，可选参数是下方的说明，如：

表 4.20 $\mathcal{AMS}$ 箭头符号

⇇	\leftleftarrows	⇉	\rightrightarrows
⇆	\leftrightarrows	⇄	\rightleftarrows
⇇	\Lleftarrow	⇉	\Rrightarrow
↞	\twoheadleftarrow	↠	\twoheadrightarrow
↢	\leftarrowtail	↣	\rightarrowtail
↫	\looparrowleft	↬	\looparrowright
⇋	\leftrightharpoons	⇌	\rightleftharpoons（重定义）
↶	\curvearrowleft	↷	\curvearrowright
↺	\circlearrowleft	↻	\circlearrowright
↰	\Lsh	↱	\Rsh
⇈	\upuparrows	⇊	\downdownarrows
↿	\upharpoonleft	↾	\upharpoonright 或 \restriction
⇃	\downharpoonleft	⇂	\downharpoonright
⊸	\multimap	⇝	\rightsquigarrow
↭	\leftrightsquigarrow	⇝	\leadsto

4-3-15

```
\[ A  \xleftarrow{0<x<1} B
  \xrightarrow[x\leq 0]{x\geq 1} C  \]
```

$$A \xleftarrow{0<x<1} B \xrightarrow[x\leq 0]{x\geq 1} C$$

extarrows 宏包使用同样的语法，将可延长的箭头扩充到了其他符号（见表 4.21）。

比这类可延长箭头更为复杂的带箭头公式，特别是交换图表的输入，参见 5.5.1 节“Xy-pic 与交换图表”。

LaTeX 定义了几个逻辑符号命令：\iff, \implies, \impliedby，它们的符号与一般的箭头相同，但间距比一般的运算符和关系符大一些，意义更为明显。amsmath 还定义了 \And 命令，也可用于逻辑表达式，如：

4-3-16

```
$x=y \implies x+a=y+a$ \\
$x=y \impliedby x+a=y+a$ \\
$x=y \iff x\le y \And x\ge y$
```

$$x=y \implies x+a=y+a$$
$$x=y \impliedby x+a=y+a$$
$$x=y \iff x\le y \And x\ge y$$

命令 \mathbin 和 \mathrel 分别用来将其参数看做是二元运算符和二元关系符。例如，$\mathcal{AMS}$ 符号 ♡ 本来只是普通符号，但可以通过临时转换类型排版下面的效果：

表 4.21　可延长的箭头符号

\xleftarrow	$A \xleftarrow[xyz]{a+b+c} Z$	\xrightarrow	$A \xrightarrow[xyz]{a+b+c} Z$
\xlongleftarrow	$A \xleftarrow[xyz]{a+b+c} Z$	\xlongrightarrow	$A \xrightarrow[xyz]{a+b+c} Z$
\xLongleftarrow	$A \xLeftarrow[xyz]{a+b+c} Z$	\xLongrightarrow	$A \xRightarrow[xyz]{a+b+c} Z$
\xleftrightarrow	$A \xleftrightarrow[xyz]{a+b+c} Z$	\xLeftrightarrow	$A \xLeftrightarrow[xyz]{a+b+c} Z$
\xlongleftrightarrow	$A \xleftrightarrow[xyz]{a+b+c} Z$	\xLongleftrightarrow	$A \xLeftrightarrow[xyz]{a+b+c} Z$
\xlongequal	$A \xlongequal[xyz]{a+b+c} Z$		

前两个由 amsmath 宏包提供，其余由 extarrows 提供。
在上下方内容很短时，带 long 或 Long 的命令比不带的更长一些。

```
运算 $\heartsuit$ 的交换律：
\[  a \mathbin{\heartsuit} b =
b \mathbin{\heartsuit} a  \]
```

运算 $\heartsuit$ 的交换律：

$$a \heartsuit b = b \heartsuit a$$

4-3-17

当然，更常见的用途是用它们制作新的符号：

```
\newcommand\varnotin{%
  \mathrel{\overline{\in}}}
$\forall x$, $\forall S$, $x\varnotin S$.
```

$\forall x, \forall S, x \varnotin S.$

4-3-18

4.3.4　括号与定界符

数学公式离不开括号的使用。括号种类大致有圆括号 ( )、方括号 [ ]、花括号 { }、尖括号 ⟨ ⟩ 等等。在 TEX 中，各色括号被分为开符号和闭符号，毫无疑问，左边的括号是开符号，右边的括号是闭符号。通常开符号左边比右边留有更大的间距，闭符号与之相反。

除了表 4.22、表 4.23 中列出的命令，amsmath 还定义了 \lvert、\rvert 和 \lVert、\rVert 分别作为 \vert 和 \Vert 对应的开符号和闭符号的命令。在表示范数、绝对值之类的概念时，它们比单纯的| 和 \| 更为明确，还可以自定义命令进一步简化代码：

4-3-19

```
\newcommand*\abs[1]{\lvert#1\rvert}
$\abs{x+y} \le \abs{x} + \abs{y}$
```

$|x+y| \le |x| + |y|$

表 4.22 LaTeX 的括号定界符

开符号		闭符号		说明
(	(	)	)	圆括号
[	[	]	]	方括号
{	\lbrace 或 \{	}	\rbrace 或 \}	花括号
⟨	\langle	⟩	\rangle	尖括号，明确指定类型时可用 < 和 >
⌊	\lfloor	⌋	\rfloor	向下取整
⌈	\lceil	⌉	\rceil	向上取整

表 4.23 LaTeX 的非括号定界符

/	/	|	\vert 或 |	↑	\uparrow	↓	\downarrow	↕	\updownarrow
\	\backslash	‖	\Vert 或 \\|	⇑	\Uparrow	⇓	\Downarrow	⇕	\Updownarrow

前两列原为普通符号，后三列原为二元关系符。

在 TeX 中的一个更为广泛的概念是数学公式中的定界符（delimiter），定界符通常就是公式两侧的括号，但也有时表示其他符号（如 $f'(x)|_{x=0}$ 中的竖线）。定界符有一个特别有用的性质，就是它可以按需要改变大小：

$$\Biggl(\biggl(\Bigl(\bigl(((\,))\bigr)\Bigr)\biggr)\Biggr) \qquad \Biggl\{\biggl\{\Bigl\{\bigl\{\{\{\,\}\}\bigr\}\Bigr\}\biggr\}\Biggr\}$$

这种可变大小的定界符主要是使用 \left 和 \right 命令得到的，它们分别把作为其参数的定界符转换为开符号和闭符号，同时使定界符可以按中间内容的高度自动调节大小，如：

4-3-20

```
\[
\partial_x \partial_y \left[
\frac12 \left( x^2+y^2 \right)^2 + xy
\right]
\]
```

$$\partial_x \partial_y \left[\frac12 \left(x^2+y^2 \right)^2 + xy \right]$$

\left 和 \right 命令必须在同一行配对，但用来配对的定界符并不需要与原来的是同一种括号，甚至可以使用一个句号 . 表示空的定界符，如：

```
\[  \left.
\int_0^x f(t,\lambda) \,\mathrm{d}t
    \right|_{x=1}, \qquad
\lambda \in
  \left[\frac12,\infty\right).  \]
```

$$\int_0^x f(t,\lambda)\,\mathrm{d}t\Big|_{x=1}, \qquad \lambda \in \left[\frac{1}{2},\infty\right).$$ 4-3-21

除了 \left 和 \right，还有一个 \middle 命令[①]，它可以在 \left 和 \right 的中间再加一个定界符，如：

```
\[
  \Pr \left( X>\frac12
  \middle\vert Y=0 \right)
= \left.
  \int_0^1 p(t)\,\mathrm{d}t
  \middle/ ( N^2+1 ) \right.
\]
```

$$\Pr\left(X > \frac{1}{2}\middle| Y = 0\right) = \int_0^1 p(t)\,\mathrm{d}t \Big/ (N^2+1)$$ 4-3-22

TEX 并不总能自动得到合适的定界符大小，也可以使用命令进行手工调整。\big, \Big, \bigg, \Bigg 分别用于将定界符放大到不同的尺寸（见表 4.24）。更常用的是在其后增加 l, r 和 m 的命令，分别表示将定界符作为开符号、闭符号和中间的二元关系符，例如：

```
\[
\biggl( \sum_{i=1}^n A_i \biggr) \cdot
\biggl( \sum_{i=1}^n B_i \biggr) > 0
\]
```

$$\left(\sum_{i=1}^{n} A_i\right) \cdot \left(\sum_{i=1}^{n} B_i\right) > 0$$ 4-3-23

```
$ 1 + \Bigl(2 - \bigl(3 \times
  (4 \div 5) \bigr) \Bigr) $
```

$$1 + \Big(2 - \big(3 \times (4 \div 5)\big)\Big)$$ 4-3-24

① 这个命令是 ε-TEX 扩展命令，不过现在几乎所有 TEX 引擎都支持 ε-TEX 扩展，可以放心使用。

表 4.24 手工调整定界符的大小

正常	(|) [‖] { }
\big \bigl \bigr \bigm	(|) [‖] { }
\Big \Bigl \Bigr \Bigm	(|) [‖] { }
\bigg \biggl \biggr \biggm	(|) [‖] { }
\Bigg \Biggl \Biggr \Biggm	(|) [‖] { }

定界符 ⟨ ⟩ 通常用 \langle 和 \rangle 得到，不过当使用 \left, \right 或 \bigl, \biggr 等命令时，也可以简单地用 < 和 > 输入，如：

4-3-25

```
\[  P = \biggl< \frac12 \biggr>, \qquad
M = \left< \begin{matrix}
  a & b \\ c & d \\
\end{matrix}\right>  \]
```

$$P = \left\langle \frac{1}{2} \right\rangle, \qquad M = \left\langle \begin{matrix} a & b \\ c & d \end{matrix} \right\rangle$$

命令 \mathopen 和 \mathclose 可以将其参数设置为开符号和闭符号，不过用途很少。声明新的可变大小的定界符需要数学字体的支持，可参见 Braams et al. [33]、LaTeX3 Project Team [142]。

4.3.5 标点

最后一类符号是数学标点，逗号就是其中的典型，它在前面一般不留间距，后面则留有小的间距。比之丰富的其他符号，数学标点只有寥寥几个[①]，见表 4.25。

键盘上的圆点 . 在数学公式中是普通符号，通常表示小数点。不过在显示公式的末尾也经常用它表示句号，在行末无须为后面的间距问题困扰。

还需要特别说明的是冒号。直接从键盘上输入 : 得到的是二元关系符，因此可以直接用 $f(x):=x^2$ 得到 $f(x) := x^2$，而无须作特别的处理（两个关系符在一起，中间没有间距），或者用它代替竖线表示集合关系，如 $\{x : x>0\}$ 得到 $\{x : x > 0\}$；由于关系符和二元运算符十分接近，也可以用它表示比例，如：

4-3-26

```
$a:b = ac:bc$
```

$$a : b = ac : bc$$

① 为了得到稍大的间距，叹号和问号实际被定义为闭符号，但仍然作为标点符号使用，它们也不是可变大小的定界符。

表 4.25　数学标点符号

名称	命令	示例
逗号	,	$f(x,y,z)=x+y+z$
分号	;	$P(a;m,n)=P(b;m,n)$
叹号	!	$\mathrm{P}_n^m=n!/(n-m)!$
问号	?	$x^2=1,x=\pm1?$
冒号	\colon	$f\colon x\mapsto x^2$

不过更准确的用法是以 \mathbin{:} 表示比例计算。区别较大的是作为标点的冒号，作为数学标点的 \colon 则在两侧有不同的间距，不能误用为：这个二元关系符，如：

```
\[  \Pr(x\colon g(x)>5) = 0.25,
  \qquad g\colon x \mapsto x^2  \]
```

$$\Pr(x\colon g(x)>5)=0.25,\qquad g\colon x\mapsto x^2$$

4-3-27

行内公式中，数学标点后面不允许断行这一事实有时可能会使人困惑。诚然，没有人希望在 $f(x,y,z)$ 的中间断行，不过如果是输入 a_1,a_2,a_3,a_4 这样的列表，又希望它可以断行，则应该把每一项放在单独的数学环境中，以文本模式的空格隔开，写成 a_1,␣a_2,␣a_2,␣a_4。这里如果需要 1, 2, ... 这样的省略号，可以用 1,␣2,\,\ldots 的方式输入。

可以使用 \mathpunct 命令把一个符号看做数学标点。事实上，\colon 命令的定义就相当于 \mathpunct{:}。类似地也有数学标点类型的数学句号 \ldotp，不过很少使用。

表 4.26　数学省略号

...	\ldots	⋯	\cdots	⋮	\vdots	⋱	\ddots	⋰	\iddots
...	\dotsc	⋯	\dotsb	⋯	\dotsm	⋯	\dotsi	...	\dotso

在 amsmath 下，\ldots 和 \ldots 在大多数情况下可直接使用 \dots 得到。
\iddots 需要 mathdots 宏包。下面一行是 amsmath 提供的细化用途的省略号。

省略号并不属于 TEX 的数学标点类型，但确是公式中常用的标点符号（见表 4.26）。在数学公式中，水平的省略号有两种，一是圆点在基线位置的 \ldots（...），二是圆点在中间位置的 \cdots（⋯）。位置较低的 \ldots 主要用在逗号之间，如 $(1,\ldots,n)$。位置在中间的 \cdots 则用在二元运算符、关系符之间，如 $1+\cdots+n$；或者表示没有乘号的连乘积，如 $a_1\cdots a_n$；或者连接多个积分号，如 $\int_0^1\cdots\int_0^1$。大体的原则是省略号与

前后的符号高度一致。使用 amsmath 时，只要用命令 \dots，在多数情况下就可以自动识别各种情形，选用正确的符号，如：

4-3-28

```
\[  (1,\dots,n) \qquad 1+\dots+n
\qquad  a=\dots=z  \]
```

$$(1,\dots,n) \qquad 1+\cdots+n \qquad a=\cdots=z$$

此外，amsmath 还按照 \dots 使用的上下文，按逗号（comma）、二元运算或关系符（binary）、乘法运算（multiplication）、积分（integral）和其他情形（other），分别定义了 \dotsc, \dotsb, \dotsm, \dotsi 和 \dotso 这几个命令。这些命令仔细设置了位置和前后间距，可以在 \dots 不能正确自动识别时使用，如连乘积与积分：

4-3-29

```
\[  \prod_{i=1}^n a_i = a_1 \dotsm a_n
\qquad \int_0^1\dotsi\int_0^1  \]
```

$$\prod_{i=1}^{n} a_i = a_1 \cdots a_n \qquad \int_0^1 \cdots \int_0^1$$

垂直和倾斜的省略号用于排版矩阵，参见 4.2.5 节。

Unicode 下的数学字体

TEX 系统的数学字体最初都是基于 METAFONT 格式的，高德纳教授最初为 TEX 设计的 Computer Modern 系列字体，就是以 METAFONT 格式制做的。随着 PostScript 和 PDF 格式在出版界的流行，各类 PostScript 字体，特别是其中的 Type 1 字体被各种 TEX 输出驱动所支持，并逐渐成为主流字体格式，原来使用 METAFONT 制做的字体，也大多使用 Type 1 格式重制。现在，新的 OpenType 字体格式随着 Unicode 字符编码正在成为新一代 TEX 引擎的标准配置，TEX 使用的字体，包括数学字体，也开始有了基于 Unicode 编码的 OpenType 格式。

TrueType 和 OpenType 格式的数学符号字体很早就已经出现，用于各种不同的数学公式排版软件中。由于各种软件使用的数学符号的数量和排列次序各不相同，公式中符号尺寸变化和相互位置等参数也主要依靠各个软件本身的功能，因此不同软件使用的数学符号字体很难相互使用，大多数排版效果也不及 TEX 系统使用原有的字体排版的结果。因此这些字体也没有被 TEX 所利用。

解决字体中符号的取舍的问题，可以依赖 Unicode 编码。但事实上，在 STIX 字体项目组的协调努力下，2002 年发布的 Unicode 3.2 版本中，才有了与 TEX 系统数量大致相当的数学字母与符号[277]。而要有包含这些符号并与

Unicode 标准相匹配的字体则更难。而且在 TEX 中，数学字体符号本身的尺寸变化、位置参数比一般字体要复杂，对应到新的 Unicode 字体中也必须要有与之相应的字体参数——这便是 OpenType 字体的数学参数表（MATH table）。这一工作是由微软公司为其 Office 2007 软件完成的[276]。微软公司不仅为 OpenType 字体格式扩展了数学参数表，同时还设计了使用这一参数表的 Cambria Math 数学字体，与 Office 新的默认西文衬线字体族 Cambria 配合，同时利用这一技术大幅加强了 Word 软件输出数学公式的能力。值得一提的是，微软在设计数学参数表时，大量借用了 TEX 中成熟的字体参数，因此在 TEX 中使用功能完整的 OpenType 数学字体也就成为可能。

事实上，unicode-math 宏包[213] 为 XeLaTeX 和 LuaLaTeX 实现了对 Unicode 编码的 OpenType 数学字体的调用。例如，要在文档中使用微软 Office 自带的 Cambria 及 Cambria Math 作为正文和数学字体，就可以使用：

```
% XeLaTeX 或 LuaLaTeX 编译
\usepackage{unicode-math}
\setmainfont{Cambria}
\setmathfont{Cambria Math}
```

在微软的 Cambria Math 之后，新的 Unicode 数学字体也不断出现：Times 风格的 TG Termes Math、XITS Math，Palatino 风格的 Asana Math、TG Pagella Math，与 LaTeX 默认字体相匹配的 Latin Modern Math，最初由 Hermann Zapf 设计的 Neo Euler 等，是目前已有的几种免费 Unicode 数学字体（见表 4.27）。此外还有 Minion Math、Lucida New Math 等几种高质量的商业字体。这些字体不仅可以用在 TEX 中，也可以安装到 Windows 操作系统中给 Office 软件使用。

使用 Unicode 数学字体，不仅可以访问丰富的符号[210]，同时可以不使用命令，直接在源文件中输入 Unicode 符号。例如使用：

```
\[
  ∫Γ(x)dx → ±∞
\]
```

来输入

$$\int \Gamma(x)dx \to \pm\infty$$

这样数学文档的源代码将会比以前变得更加易读。

表 4.27 免费或容易获得的 Unicode 数学字体

数学字体	匹配的正文字体	示例
Asana Math	Palatino	We have formula $\int_{z_0}^{z_1} \cos(2\pi z)\,\mathrm{d}z.$
Latin Modern Math	Latin Modern Roman	We have formula $\int_{z_0}^{z_1} \cos(2\pi z)\,\mathrm{d}z.$
Neo Euler	Palatino 等 †	We have formula $\int_{z_0}^{z_1} \cos(2\pi z)\,\mathrm{d}z.$
TG Pagella Math	TeX Gyre Pagella	We have formula $\int_{z_0}^{z_1} \cos(2\pi z)\,\mathrm{d}z.$
TG Termes Math	TeX Gyre Termes	We have formula $\int_{z_0}^{z_1} \cos(2\pi z)\,\mathrm{d}z.$
XITS Math	XITS	We have formula $\int_{z_0}^{z_1} \cos(2\pi z)\,\mathrm{d}z.$
Cambria Math	Cambria	We have formula $\int_{z_0}^{z_1} \cos(2\pi z)\,\mathrm{d}z.$

†Neo Euler 是 Zapf 为 AMS 设计的 Euler 字体的 Unicode 版本，它本身并非为了配合具体某种正文字体设计，不过最早用于和 Concrete 字体一起排版《具体数学》一书。

4.4 多行公式

如果所有的数学公式都像 $2+2=4$ 那么简单该有多好！可是事与愿违，实际中经常会碰到长得一行写不下的公式、跨越几行的数学结构或者延伸多行的演算列表，比如容斥原理：

$$\begin{aligned}|A_1 \cup A_2 \cup A_3 \cdots \cup A_n| = &\sum_{1\le i_1\le n} |A_{i_1}| - \sum_{1\le i_1<i_2\le n} |A_{i_1} \cap A_{i_2}| \\ &+ \sum_{1\le i_1<i_2<i_3\le n} |A_{i_1} \cap A_{i_2} \cap A_{i_3}| + \cdots \\ &+ (-1)^{k-1} \sum_{1\le i_1<\cdots<i_k\le n} |A_{i_1} \cap \cdots \cap A_{i_k}| + \cdots \\ &+ (-1)^{n-1} |A_1 \cap \cdots \cap A_n|\end{aligned} \tag{4.8}$$

或是一个三元的线性方程组：

$$a_{11}x + a_{12}y + a_{13}z = A \tag{4.9a}$$

$$a_{21}x + a_{22}y + a_{23}z = B \tag{4.9b}$$

$$a_{31}x + a_{32}y + a_{33}z = C \tag{4.9c}$$

式 (4.8) 是把一个长公式拆开来，整个公式仍是一体（从编号就可看出），而式 (4.9) 则是将三个方程分别列出，每个公式相互联系而又各自独立。还可能出现更复杂的情况，比如把一组多行的公式作为大公式的一部分：

$$f(x) \triangleq \begin{cases} \frac{3}{2} - x, & \frac{1}{2} < x \leq 1, \\ \frac{3}{4} - x, & \frac{1}{4} < x \leq \frac{1}{2}, \\ \frac{3}{8} - x, & \frac{1}{8} < x \leq \frac{1}{4}, \\ \vdots & \vdots \end{cases} \tag{4.10}$$

下面我们将把注意力主要放在显示公式上，研究输入多行公式的各种情形。除了 amsmath，这一节我们也会看到许多来自 mathtools 宏包的环境和命令。这些宏包的功能都比较芜杂，更详细的内容读者可参考宏包文档 American Mathematical Society [7]、Høgholm [110]。

4.4.1　罗列多个公式

把几行公式罗列在一起，是基本的产生多行公式的方法。标准 LaTeX 为此提供了 eqnarray 和 eqnarray* 环境，用来输入按等号（或其他关系符）对齐的方程组，不过它在等号两边留的间距过大，使用也不大方便，因此我们最好只把它留给历史文档[①]。使用更为灵活的是由 amsmath 提供的一系列数学环境，它们可以罗列各种对齐或不对齐的公式组。

如果问什么是显示方程的基本输入方法，那么回答便是使用编号的 equation 环境和不编号的 equation* 环境（\[和 \] 是它的简写），但是里面的换行命令 \\ 是无效的。输入多行数学公式最基本的方法与它如出一辙，就是使用 gather 和 gather*（不编号）环境，不同的仅仅是可以使用 \\ 进行换行，如：

① eqnarray 环境用起来像是一个有三列的表格，表列格式是 rcl，带星号的环境不编号，如果有需要，可参考其他书籍或文档。

4-4-1

```
\begin{gather}
a+b = b+a \\
ab = ba
\end{gather}
```

$$a+b=b+a \tag{4.11}$$
$$ab=ba \tag{4.12}$$

4-4-2

```
\begin{gather*}
3+5 = 5+3 = 8 \\
3\times 5 = 5\times 3
\end{gather*}
```

$$3+5=5+3=8$$
$$3\times 5=5\times 3$$

在编号的多行公式中，可以在这一行 \\ 之前使用 \notag 命令阻止指定的行不编号，如：

4-4-3

```
\begin{gather}
3^2 + 4^2 = 5^2 \notag \\
5^2 + 12^2 = 13^2 \notag \\
a^2 + b^2 = c^2
\end{gather}
```

$$3^2+4^2=5^2$$
$$5^2+12^2=13^2$$
$$a^2+b^2=c^2 \tag{4.13}$$

gather 环境得到的公式是每行居中的，align 环境则允许公式按等号或其他关系符对齐，在关系符前加 & 表示对齐。align* 功能相同，但不编号。例如：

4-4-4

```
\begin{align}
  x &= t + \cos t + 1 \\
  y &= 2\sin t
\end{align}
```

$$x=t+\cos t+1 \tag{4.14}$$
$$y=2\sin t \tag{4.15}$$

align 和 align* 环境还允许排列多列对齐的公式，列与列之间仍使用 & 分隔：

4-4-5

```
\begin{align*}
x &= t   & x &= \cos t     & x &= t \\
y &= 2t & y &= \sin(t+1) & y &= \sin t
\end{align*}
```

$$\begin{aligned} x&=t & x&=\cos t & x&=t \\ y&=2t & y&=\sin(t+1) & y&=\sin t \end{aligned}$$

需要注意的是，align 环境中的列分隔符 & 一般应该放在二元关系符的前面，这样才能保证正确的符号间距（有时也可以放在二元运算符前面）。如果个别需要在关系符后面或其他地方对齐的，则应该注意使用的符号类型，或者使用 phantom 等命令变通一下。例如，如果要得到公式

$$\begin{aligned}&(a+b)(a^2-ab+b^2)\\&=a^3-a^2b+ab^2+a^2b-ab^2+b^2\\&=a^3+b^3\end{aligned}\tag{4.16}$$

则可以使用图 4.1 中的任一种方式。

4-4-6

```
% 关系符后对齐，需要使用空的分组
% 代替关系符右侧符号，保证间距
\begin{align*}
    & (a+b)(a^2-ab+b^2) \notag \\
={} & a^3 - a^2b + ab^2 + a^2b
     - ab^2 + b^2 \notag \\
={} & a^3 + b^3 \label{eq:cubesum}
\end{align*}
```

4-4-7

```
% 缺少关系符，需要使用幻影给关系符
% 占位，并利用 \mathrel 保证间距
\begin{align*}
&\mathrel{\phantom{=}}
   (a+b)(a^2-ab+b^2) \notag \\
&= a^3 - a^2b + ab^2 + a^2b
   - ab^2 + b^2 \notag \\
&= a^3 + b^3 \label{eq:cubesum}
\end{align*}
```

图 4.1　(4.16) 式中对齐方式特殊的 align 环境写法

flalign 环境与 flalign* 环境的功能与 align 环境类似，它也把公式每两列分为一组，分别向右向左对齐。不同的是，flalign 环境中公式的间距是可以无限伸长的弹性距离，几列公式会在水平方向分散对齐，如：

```
\begin{flalign}
  x &= t  & x &= 2 \\
  y &= 2t & y &= 4
\end{flalign}
```

4-4-8

$$x=t \qquad x=2 \tag{4.17}$$

$$y=2t \qquad y=4 \tag{4.18}$$

alignat 环境则与 flalign 正相反，它本身不在列与列之间产生间距，但可以手工增加间距。alignat 环境有一个参数，表示每行要对齐的公式个数（每两列一组）。如下面把列间距设定为一个 \quad 的距离（即 1 em）：

4-4-9

```
\begin{alignat}{2}
  x &= \sin t &\quad&\text{水平方向
} \\
  y &= \cos t &&\text{垂直方向}
\end{alignat}
```

$$x = \sin t \quad \text{水平方向} \tag{4.19}$$
$$y = \cos t \quad \text{垂直方向} \tag{4.20}$$

也可以用 alignat 环境代替频繁的 \phantom，产生一些特别的对齐效果：

4-4-10

```
\begin{alignat*}{6}
 &1 & &+2 & &+3 & &+4 & &+5 & &=15 \\
 &1 & &   & &+3 & &   & &+5 & &=9 \\
 &  & &+2 & &   & &+4 & &   & &=6
\end{alignat*}
```

$$\begin{aligned} 1 &+ 2 + 3 + 4 + 5 = 15 \\ 1 &\quad + 3 \quad + 5 = 9 \\ &+ 2 \quad + 4 \quad = 6 \end{aligned}$$

有时，我们需要在几行公式中插入简短的文字，同时又不想破坏公式的对齐，这时可以使用 \intertext 命令。\intertext 本身可以表示换行，因而前面一行的 \\ 可以省略，如：

4-4-11

```
\begin{align*}
x^2 + 2x &= -1
\intertext{移项得}
x^2 + 2x + 1 &= 0
\end{align*}
```

$$x^2 + 2x = -1$$

移项得

$$x^2 + 2x + 1 = 0$$

也可以用 mathtools 提供的 \shortintertext 命令代替 \intertext，以得到更为紧凑的行间距。

在罗列多行公式时，经常需要使用像式 (4.9) 那样的几个子公式编号，这可以在数学环境外套一层 subequations 环境得到。使用 subequations 环境，则环境内的多个公式都被看做一个公式，使用相同的一个主编号，里面分别再按字母对子公式进行编号。subequations 环境和里面的子公式可以分别使用 \label 设定标签引用。下面举一个比较复杂的例子：

```
设 $G$ 是一个带有运算 $*$ 的集合，则 $G$ 是\emph{群}，当且仅当：
\begin{subequations}\label{eq:group}
\begin{alignat}{2}
```

```
\forall a,b,c &\in G, &\qquad (a*b)*c &= a*(b*c);\label{subeq:assoc}\\
\exists e, \forall a &\in G, &  e*a &= a; \\
\forall a, \exists b &\in G, &  b*a &= e.
\end{alignat}
\end{subequations}
式~\eqref{eq:group} 的三个条件中，\eqref{subeq:assoc}~又称为结合律。
```

4-4-12

设 G 是一个带有运算 $*$ 的集合，则 G 是群，当且仅当：

$$\forall a,b,c \in G, \qquad (a*b)*c = a*(b*c); \tag{4.21a}$$

$$\exists e, \forall a \in G, \qquad e*a = a; \tag{4.21b}$$

$$\forall a, \exists b \in G, \qquad b*a = e. \tag{4.21c}$$

式 (4.21) 的三个条件中，(4.21a) 又称为结合律。

练 习

4.9 排版第 263 页式 (4.9)。

4.4.2 拆分单个公式

研究完多个公式的罗列，现在来看如何把一个公式拆成好几行。当然，这个工作很多时候似乎也可以使用 align 或 align* 完成，不过如果公式需要单个编号，那么最好还是使用单独的命令。

multline 环境和 multiline* 环境是 equation 环境的分行版本，它可以使用 \\ 换行。各行对齐的方式是：第一行左对齐，最后一行右对齐，中间的部分居中。不过左右两边与版心边界都留有一小段间距。这种环境特别适合排版非常长的连续运算，如：

```
\begin{multline}
a+b+c+d+e \\
 +f+g+h+i+j \\
 +k+l+m+n+o \\
   +p+q+r+s+t
\end{multline}
```

$$\begin{multline} a+b+c+d+e \\ +f+g+h+i+j \\ +k+l+m+n+o \\ +p+q+r+s+t \end{multline} \tag{4.22}$$

4-4-13

multline 环境首尾两行与版心边界的间距分别由长度变量 \multlinegap 和 \multlinetaggap 控制[①]。同时也可以使用 \shoveleft 和 \shoveright 命令指定中间的行像首尾两行一样左对齐或右对齐，如：

4-4-14

```
\setlength{\multlinegap}{3em}
\setlength{\multlinetaggap}{3em}
\begin{multline*}
1+2+3 \\ \shoveleft{+4+5+6} \\
+7+8+9 \\
\shoveright{+10+11+12} \\ +13+14+15
\end{multline*}
```

$$1+2+3$$
$$+4+5+6$$
$$+7+8+9$$
$$+10+11+12$$
$$+13+14+15$$

除了 multline 环境，更常用的可能就是 split 环境了。split 环境并不开始一个数学公式，它用在 equation、gather 等数学环境里面，可以把单个公式拆分成多行，同时支持类似 align 环境那样的对齐方式（但不能对齐多列公式），例如：

4-4-15

```
\begin{equation}  \begin{split}
\cos 2x &= \cos^2 x - \sin^2 x \\
        &= 2\cos^2 x - 1
\end{split}  \end{equation}
```

$$\begin{aligned}\cos 2x &= \cos^2 x - \sin^2 x \\ &= 2\cos^2 x - 1\end{aligned} \tag{4.23}$$

split 环境与 align 的最大区别就是 split 环境不产生编号，编号仍然由外面的数学环境产生，因此两行的公式产生的编号仍在两行中间；而 align 环境本身就是数学环境，它会给每一行公式产生一个编号。

同样，如果 split 环境中的某一行不是在二元关系符前面对齐，通常需要另行设置间距或对齐方式，如：

```
\begin{equation}\label{eq:trigonometric}
\begin{split}
\frac12 (\sin(x+y) + \sin(x-y))
  &= \frac12(\sin x\cos y + \cos x\sin y) \\
  &\quad + \frac12(\sin x\cos y - \cos x\sin y) \\
```

① 其中变量 \multlinetaggap 控制的是可能有编号的一行的间距（包括编号的宽度），与编号在左在右有关，默认值是 10pt。在 amsmath 的文档 American Mathematical Society [7] 中，缺少对命令 \multlinetaggap 的说明，这里根据实际源代码增加。

```
  &= \sin x\cos y
\end{split}
\end{equation}
```

4-4-16

$$\begin{aligned}\frac{1}{2}(\sin(x+y)+\sin(x-y)) &= \frac{1}{2}(\sin x\cos y+\cos x\sin y)\\ &\quad+\frac{1}{2}(\sin x\cos y-\cos x\sin y)\\ &= \sin x\cos y\end{aligned} \tag{4.24}$$

在输入了十几个这样折行的公式以后，你或许会对原来 split 环境“在合适的地方插入 & 和 \\”感到厌倦。可能必须要编译过一遍才知道一个公式是在一行可以排下还是必须另起一行，而当文档要从单栏变成双栏时，调整换行的问题会变得更加严重。幸好，breqn 宏包[109] 可以帮助我们自动对长的显示公式折行。breqn 宏包主要提供了 dmath 和 dmath* 等几个环境，产生可以自动折行的显示公式。如用下面的代码也可以得到与式 (4.24) 相同的换行、对齐效果，且会随行宽而自动调整折行的位置，非常方便：

```
% \usepackage{breqn}
\begin{dmath}\label{eq:trigonometric}
\frac12 (\sin(x+y) + \sin(x-y)) = \frac12(\sin x\cos y + \cos x\sin y)
+ \frac12(\sin x\cos y - \cos x\sin y) = \sin x\cos y
\end{dmath}
```

4-4-17

需要注意的是，在 dmath 等环境中使用一些简单的上下标（如 N^+）时也会要求必须使用分组（即 N^{+}）。另外，直接使用 breqn，可能会与少量有关上下标的宏包（如 CJK 宏包，以及前面介绍的 tensor）发生冲突，此时需要给它加上 mathstyleoff 选项改用较安全的模式工作，因此应该小心选用，在用 \usepackage 调用宏包时也应该尽量放在导言区的后面。有关 breqn 宏包更多的用法和注意事项，可参见文档 Høgholm [109]。

练 习

4.10 排版第 262 页式 (4.8) 的容斥原理。

4.4.3 将公式组合成块

下面我们来看如何把几行公式组合在一起来成为更大的公式的一部分。

最为常见的是 cases 环境，它在几行公式前面用花括号括起来，用来表示几种不同的情况。每行公式中使用 & 分隔为两部分，通常表示值和后面的条件，例如：

4-4-18
```
\begin{equation}\label{eq:dirichlet}
D(x) = \begin{cases}
1, & \text{if } x \in \mathbb{Q}; \\
0, & \text{if } x \in
    \mathbb{R}\setminus\mathbb{Q}.
\end{cases}
\end{equation}
```

$$D(x) = \begin{cases} 1, & \text{if } x \in \mathbb{Q}; \\ 0, & \text{if } x \in \mathbb{R} \setminus \mathbb{Q}. \end{cases} \tag{4.25}$$

mathtools 宏包则进一步提供了 dcases 环境，它保证每行公式都是显示格式的大小（\displaystyle，参见 4.5.2 节），如：

4-4-19
```
% \usepackage{mathtools}
\[ \left\lvert x - \frac12 \right\rvert
= \begin{dcases}
x-\frac12,  & x \geq \frac12;\\
\frac12-x,  & x < \frac12.
\end{dcases}  \]
```

$$\left| x - \frac{1}{2} \right| = \begin{cases} x - \frac{1}{2}, & x \geq \frac{1}{2}; \\ \frac{1}{2} - x, & x < \frac{1}{2}. \end{cases}$$

想在 cases 环境的每行公式后面增加编号？没问题，cases 宏包[13] 的 numcases 环境正是你所需要的。numcases 环境本身就是一个数学环境[①]，它具有特别的语法：

\begin{numcases}{⟨左边的子公式⟩**}**
⟨子公式一⟩ & ⟨条件一⟩ \\
⟨子公式二⟩ & ⟨条件二⟩ \\
……
\end{numcases}

但注意这里的 ⟨条件⟩ 是在文本模式而不是数学模式下的，例如：

```
% \usepackage{cases}
\begin{numcases}{f(x)=}
  1/q, & if $x = p/q \in \mathbb{Q}$; \\
```

① 因此，从用法上说，它其实应该放在 4.4.1 节。

```
  0,    & else.
\end{numcases}
```

4-4-20

$$f(x)=\begin{cases}1/q, & \text{if } x=p/q\in\mathbb{Q};\quad (4.26)\\ 0, & \text{else.}\quad (4.27)\end{cases}$$

cases 环境的功能太过特殊，下面来看几个更一般的组合公式块的环境。amsmath 用来罗列多个公式的环境，大都效果类似的对应组合公式块的环境，它们的环境名通常是在原来的环境名的动词后面加 ed，包括 gathered 环境、aligned 环境和 alignedat 环境等。

gathered 环境把几行公式居中排列，组合为一个整体，如：

```
\[  \left. \begin{gathered}
  S \subseteq T \\
  S \supseteq T
\end{gathered} \right\}
\implies S = T  \]
```

$$\left.\begin{gathered}S\subseteq T\\ S\supseteq T\end{gathered}\right\}\Longrightarrow S=T$$

4-4-21

类似地，mathtools 宏包提供的 lgathered 和 rgathered 环境则把几行公式向左、向右对齐排列，组合为一个整体，如：

```
% \usepackage{mathtools}
\[  \text{比较曲线}
\left\{ \begin{lgathered}
  x = \sin t, y = \cos t \\
  x = t + \sin t, y = \cos t
\end{lgathered} \right.  \]
```

$$\text{比较曲线}\begin{cases}x=\sin t, y=\cos t\\ x=t+\sin t, y=\cos t\end{cases}$$

4-4-22

同样不难想象，aligned 环境和 alignedat 环境的用法就同 align 与 alignat 命令如出一辙，如：

```
\begin{equation}\label{eq:trinary}
\begin{aligned} x+y &= -1 \\ x+y+z &= 2 \\ xyz &= -6 \end{aligned}
\implies
\begin{aligned} x+y &= -1 \\ xy &= -2 \\ z &= 3 \end{aligned}
```

```
\implies
\begin{alignedat}{3}
            x &= 1,  &\quad y &= -2, &\quad z &= 3 \\
\text{或\ }  x &= -2, &      y &= 1,  &      z &= 3
\end{alignedat}
\end{equation}
```

4-4-23

$$\begin{array}{l} x+y=-1 \\ x+y+z=2 \\ xyz=-6 \end{array} \Longrightarrow \begin{array}{l} x+y=-1 \\ xy=-2 \\ z=3 \end{array} \Longrightarrow \begin{array}{l} x=1, \quad y=-2, \quad z=3 \\ \text{或}\, x=-2, \quad y=1, \quad z=3 \end{array} \tag{4.28}$$

mathtools 宏包还提供了 multlined 环境，可以把折行的长公式也作为一个块使用，如：

```
% \usepackage{mathtools}
\newcommand\Set[2]{%
  \left\{#1\ \middle\vert\ #2 \right\}}
\[  \Omega = \Set{x}{\begin{multlined}
x^7+x^6+x^5 \\ +x^4+x^3+x^2 \\ +x+1=0
\end{multlined}}  \]
```

4-4-24

$$\Omega = \left\{ x \;\middle|\; \begin{array}{l} x^7+x^6+x^5 \\ +x^4+x^3+x^2 \\ +x+1=0 \end{array} \right\}$$

上面所使用的 gathered、aligned、alignedat、multlined 等环境，产生的公式块在垂直方向默认是居中对齐的（见式 (4.28)）。可以在环境后面加 [t] 或 [b] 参数，得到在第一行或最后一行对齐的公式块，例如：

```
\begin{align*}
2^5 &= (1+1)^5 \\
&= \begin{multlined}[t]
\binom50\cdot 1^5 + \binom51\cdot 1^4 \cdot 1
  + \binom52\cdot 1^3 \cdot 1^2 \\
+ \binom53\cdot 1^2 \cdot 1^3 + \binom54\cdot 1 \cdot 1^4
  + \binom55\cdot 1^5
\end{multlined} \\
&= \binom50 + \binom51 + \binom52 + \binom53 + \binom54 + \binom55
\end{align*}
```

4-4-25

$$\begin{aligned}2^5 &= (1+1)^5 \\ &= \binom{5}{0}\cdot 1^5 + \binom{5}{1}\cdot 1^4\cdot 1 + \binom{5}{2}\cdot 1^3\cdot 1^2 \\ &\quad + \binom{5}{3}\cdot 1^2\cdot 1^3 + \binom{5}{4}\cdot 1\cdot 1^4 + \binom{5}{5}\cdot 1^5 \\ &= \binom{5}{0} + \binom{5}{1} + \binom{5}{2} + \binom{5}{3} + \binom{5}{4} + \binom{5}{5}\end{aligned}$$

练 习

4.11 排版第 263 页的式 (4.10)，此式给出了区间 (0, 1] 到 (0, 1) 的一个双射。

4.12 split 环境与 aligned 环境有什么相同的地方？又有什么区别？应该如何选用这两个环境？

4.5　精调与杂项

在本章的最后，我们来讨论如何对公式的一些细节进行调整，并补充一些前面漏掉的内容。TEX 不是数学家，它不能理解公式的内容，因而可能无法得到正确的公式格式，需要我们指明输出方式；但另一方面，TEX 预设的公式格式细节比大多数未经专门研究的人都要多，所以本章的每节前面都加上了“危险”符号，以示应该慎重对待。

4.5.1　公式编号控制

显示公式的编号通常在公式右边、垂直正中的位置。不过正如 4.1 节所说的，可以使用全局的 leqno、reqno 选项控制公式编号在公式的左边或右边，也可以用 amsmath 的 centertags、tbtags 选项编号的垂直位置。在左侧的编号常常与 fleqn 选项一起使用，因此，如果使用了

```
\documentclass[fleqn,leqno]{article}
\usepackage[tbtags]{amsmath}
```

4-5-1

*本节内容初次阅读可略过。

则 split 环境得到的式 (4.24) 就会变成是这样的：

$$\begin{aligned}(4.24)\qquad \frac{1}{2}(\sin(x+y)+\sin(x-y)) &= \frac{1}{2}(\sin x\cos y+\cos x\sin y)\\ &\quad+\frac{1}{2}(\sin x\cos y-\cos x\sin y)\\ &= \sin x\cos y\end{aligned}$$

4.4.1 节我们提到使用 \notag 命令可以临时取消一行公式的编号，同样，amsmath 还提供了 \tag 命令进行手工编号，参数是文本模式的，如：

4-5-2

```
\begin{equation*}
  a^2 + b^2 = c^2 \tag{$\star$}
\end{equation*}
```

$$a^2+b^2=c^2 \qquad (\star)$$

加星号的 \tag* 命令则可以去掉公式编号原有的括号，获得最大的灵活性[①]：

4-5-3

```
\begin{equation*}
\sum_{k=1}^n \frac1k
  = \ln n + \mathrm{C} \tag*{[Euler]}
\end{equation*}
```

$$\sum_{k=1}^{n}\frac{1}{k}=\ln n+\mathrm{C} \qquad [\text{Euler}]$$

\tag 命令只是临时地修改个别的编号，如果要统一修改公式的编号格式，可以使用 mathtools 宏包的 \newtagform, \renewtagform 和 \usetagform 命令。可以用它们定义新类型的编号风格（或重定义已有类型），在文档中分别选用。这种用法和用 \pagestyle 选用不同页面风格十分类似，其语法格式如下：

\newtagform{⟨名称⟩**}[**⟨内格式⟩**]{**⟨左⟩**}{**⟨右⟩**}**
\renewtagform{⟨名称⟩**}[**⟨内格式⟩**]{**⟨左⟩**}{**⟨右⟩**}**
\usetagform{⟨名称⟩**}**

⟨名称⟩ 是所定义或使用的编号风格名，默认的风格名为 default，也就是编号在圆括号之中的形式，可选的 ⟨内格式⟩ 是带有一个参数的命令，⟨左⟩ 和 ⟨右⟩ 分别是在编号左右的括号。我们可以如下定义在方括号中斜体的格式：

① 这里两个例子也可以使用 \[和 \]，这在 amsmath 中是等价的。不过个别宏包（如 ntheorem）会另行对 \[和 \] 重定义，则会造成问题，此时需要使用 equation* 等环境。

```
\newtagform{bracket}[\textit]{[}{]}
\usetagform{bracket}
\begin{equation}
\sum_{k=1}^n \frac1k = \ln n + \mathrm{C}
\end{equation}
```

$$\sum_{k=1}^{n} \frac{1}{k} = \ln n + \mathrm{C} \qquad [4.29]$$

4-5-4

控制编号，最重要的方式当然还是计数器。公式编号的计数器是 equation，通过重定义 \theequation 可以改变编号的数字形式，例如使用“章编号”+“罗马数字编号”：

```
\renewcommand\theequation{%
  \thechapter.\roman{equation}}
\begin{equation}\label{eq:euler}
  \chi = V + F - E = 2
\end{equation}
```

$$\chi = V + F - E = 2 \quad (4.\mathrm{xxx})$$

4-5-5

如果使用子公式，其编号格式是在 subequations 环境内部设定的，此时子公式编号的计数器是 equation，父公式计数器是 parentequation。修改子公式的编号格式也必须在 subequations 环境内部，因而最好定义一个新的子公式环境，例如让子公式以罗马数字编号：

```
\newenvironment{mysubeqn}%
  {\begin{subequations}
    \renewcommand\theequation{\theparentequation-\roman{equation}}}%
  {\end{subequations}}
\begin{mysubeqn}
\begin{gather}
  \zeta(2) = \frac{\uppi^2}{6} \\
  \zeta(s) = \prod_{p\text{ prime}} \frac{1}{1 - p^{-s}}
\end{gather}
\end{mysubeqn}
```

4-5-6

$$\zeta(2) = \frac{\pi^2}{6} \tag{4.31-i}$$

$$\zeta(s) = \prod_{p \text{ prime}} \frac{1}{1 - p^{-s}} \tag{4.31-ii}$$

在标准文档类 book 和 report 中，公式是按章编号的，即每过一章（chapter）计数器 equation 清零一次。正如第 99 页 2.2.3.2 节所说，可以使用 amsmath 的 \numberwithin 命令让 article 文档类按节编号，也可以用 chngcntr 宏包取消这种计数器关联，如取消公式编号与章的关联：

4-5-7
```
\documentclass{book}
\usepackage{chngcntr}
\counterwithout{equation}{chapter}
```

练 习

4.13 使用 \tag 命令可以给公式手工添加编号，但不能自动计数。试定义一个 \addtag 命令，它的功能是在不编号的公式后面添加一个自动编号，然后使用这个命令和 align* 环境排版下面的公式：

$$\begin{aligned} A(n) &= 1+2+\cdots+n \\ &= \frac{1}{2}((1+2+\cdots+n)+(n+(n-1)+\cdots+1)) \\ &= \frac{1}{2}n(n+1) \end{aligned} \tag{4.32}$$

你的定义能正确处理交叉引用吗？

4.5.2 公式的字号

数学公式的字号受数学环境外正文字号控制。要得到巨大的公式，可以在数学环境外添加一个分组设置：

4-5-8
```
{\Large\[F(x) \equiv 0\]}
```

$$F(x) \equiv 0$$

要将这一技术用在公式内部，就需要在数学模式和文本模式下来回切换，4.2.5 节的分块矩阵就是一例，下面是一个更复杂的例子：

*本节内容初次阅读可略过。

```
\newcommand\D{\displaystyle}
\[  \mathop{\text{\Large$\D\sum_i$}}
  \dfrac{\D\int f_i(x)\,\mathrm{d}x}
    {\D\oint g_i(x)\,\mathrm{d}x}  \]
```

$$\sum_i \frac{\displaystyle\int f_i(x)\,\mathrm{d}x}{\displaystyle\oint g_i(x)\,\mathrm{d}x}$$

4-5-9

在数学公式内部，符号的大小则是按符号所处的数学结构确定的。如表 4.28 所示，在 TEX 中共有 4 种不同符号尺寸，每个尺寸又分为普通和受限两种形式。分式的分母、根式、下标、\overline、数学重音等数学结构都会使公式处于受限模式，在这种模式下符号的尺寸没有变化，但符号的上标位置会比不受限的形式低一些，以免与数学结构冲突。通常 LATEX 会自动按照公式的结构选取合适的字号，但在输入一些复杂的结构时，偶尔也可以像例 4-5-9 那样手工进行调整。

表 4.28　TEX 中数学公式的字号

记号	尺寸	用途	命令	字样
D, D'	显示尺寸	显示公式	\displaystyle	$\displaystyle\sum A(n)$
T, T'	正文尺寸	正文公式	\textstyle	$\textstyle\sum A(n)$
S, S'	标号尺寸	一级上下标	\scriptstyle	$\scriptstyle\sum A(n)$
SS, SS'	小标号尺寸	二级以上的上下标	\scriptscriptstyle	$\scriptscriptstyle\sum A(n)$

带 ′ 标记的字号是受限的形式，用在下标、分母、根式和数学重音等场合，符号大小相同，但其上标排列得更为紧凑。

在四个不同的数学尺寸中，D 和 T 是大部分符号使用相同的尺寸，只有大小可变的巨算符会有所不同，因而实际上只有三个不同的数学尺寸。文本模式的字号命令会同时影响这三个尺寸，但有时可能希望特别修改数学符号尺寸，特别是修改上下标的尺寸，此时可以使用 \DeclareMathSizes 命令声明数学字号，或用 \defaultscriptratio 和 \defaultscriptscriptratio 声明标号尺寸与正文尺寸的比例。如用下面的设置可以减小中文五号字和非标准字号角标的符号尺寸：

```
% 单位 pt，中文五号字（10.5 bp）
\DeclareMathSizes{10.54}{10.54}{6.32}{4.22}
% 默认标号尺寸是正文 0.6 倍
\renewcommand\defaultscriptratio{0.6}
% 默认小标号尺寸是正文 0.4 倍
\renewcommand\defaultscriptscriptratio{0.4}
```

4-5-10

这里，\DeclareMathSizes 的四个参数分别是文本尺寸、数学基本尺寸、数学标号尺寸、数学小标号尺寸，单位是 pt，通常前两个尺寸是相同的。在一个文本尺寸已经由 \DeclareMathSizes 声明过时，\defaultscriptratio 和 \defaultscriptscriptratio 不起作用。在 ctex 宏包及文档类中，已经为汉字的各个字号用 \DeclareMathSizes 命令设置了对应的标号尺寸，因此如果要修改标号的尺寸，必须一一用 \DeclareMathSizes 对应进行修改（可参见 ctex 宏包源代码）。

4.5.3 断行与数学间距

显示数学公式的分行我们已经在 4.4 节专门说明过，下面来看行内公式的断行。行内数学公式的断行有如正文中用连字符断词，只允许在特定的位置断开。事实上，TEX 只允许在二元关系符和二元运算符后进行断行，并且更倾向于在二元关系符后断行。因此，出现在行末的公式 $f(a,b)=a+b$ 通常总会在等号后面断开，如果位置合适也可能在加号后面断开，但决不会在逗号、括号或字母后面断开。

TEX 公式的断行规则提醒我们在写较长的连乘积时，最好带上乘号，用 × 或 · 都可以。例如，以 \cdot 分隔的 $F(x)\cdot G(x)\cdot H(x)$ 会比 $F(x)G(x)H(x)$ 更不容易产生糟糕的段落。在正文中可以使用 \- 来提示可能的断词点，对应地，在行内数学公式中也可以使用命令 * 来提示可能的乘法断点，它允许 TEX 在 * 处断行，并在断行时在行末插入一个乘号 ×。* 命令可以让 $F(x,y,z)G(x,y,z)$ 在同一行排版时不插入多余的乘号，只在它们分成两行时产生这个乘号，非常方便：

4-5-11

```
\fbox{\parbox{17em}{%
$F(x,y,z)\* G(x,y,z)$ 不同于 $F(x,y,z)\* G(x,y,z)$ 吗？}}
```

$F(x,y,z)G(x,y,z)$ 不同于 $F(x,y,z)\times G(x,y,z)$ 吗？

一个困扰很多人的断行问题是长的列表项，比如 a,b,c，它们应该分开写成 a，b，c，因为公式中逗号后面是不允许断行的，同样的规则也适用于其他数学标点。这么做会破坏 \left, \right 控制的定界符，不过倘若公式已经复杂到真的需要使用可伸缩的括号，最好的办法是使用显示公式：

*本节内容初次阅读可略过。

```
\[  \left\{
  0, 1, -1, 2, -2, \frac12, -\frac12, \frac13, -\frac13, \dotsc
\right\}  \]
```

4-5-12

$$\left\{0, 1, -1, 2, -2, \frac{1}{2}, -\frac{1}{2}, \frac{1}{3}, -\frac{1}{3}, \dots\right\}$$

你或许喜欢 * 命令的工作方式，却不喜欢它插入的 × 乘号。在 Braams et al. [33] 中可以查到这个命令的原始定义：

```
\def\*{\discretionary{\thinspace\the\textfont2\char2}{}{}}
```

其中 \def 是 \newcommand 的原始形式，而 \the\textfont2\char2 就是符号 \times（×）的原始命令写法，不必理会它，我们可以直接把它换成喜欢的乘号，放在一个盒子里面[①]：

```
\renewcommand\*{%
  \discretionary{\,\mbox{$\cdot$}}{}{}}
\fbox{\parbox{17em}{%
$F(x,y,z)\* G(x,y,z)$ 不同于 $F(x,y,z)\* G(x,y,z)$}}
```

4-5-13

$F(x,y,z)G(x,y,z)$ 不同于 $F(x,y,z)\cdot G(x,y,z)$

与断行对应的是断页。LaTeX 默认禁止在多行公式中分页，使用 \allowdisplaybreaks 命令则可以允许这种分页，这对于演算中可能出现的超长连等式是非常有用的。在多行公式中还可以使用 \displaybreak 手工分页。

下面我们来看 LaTeX 中的数学间距。数学模式有自己的数学间距命令。数学间距使用的长度单位比较特殊，记为 mu，即数学单位（math unit）。一个数学单位相当于 1/18 个 em 的长度。类似 \hspace，可以直接使用 \mspace 命令指定一定长度的间距，如 \mspace{9mu} 就是半身（0.5 em）的间距。

更为常用的则是数学模式的（负）细间距、中间距和厚间距，特别是正负的细间距：

\,	3mu	→|←
\: 或 \>	4mu plus 2mu minus 4mu	→|←
\;	5mu plus 5mu	→|←
\!	-3mu	→|←

① 有关 \discretionary 等命令，可参见 Knuth [126]。

它们通常用来对公式进行细节上的微调，或者定义新的命令，例如：

4-5-14

```
\begin{align*}
& \int f(x)\,\mathrm{d}x\,\mathrm{d}y \\
& \sqrt2 \, x    && \sqrt{\,\log x} \\
& x^2 \! / 2     && |\!{\gets} 5 {\to}\!|
\end{align*}
```

$$\int f(x)\,\mathrm{d}x\,\mathrm{d}y$$
$$\sqrt{2}\,x \qquad \sqrt{\log x}$$
$$x^2/2 \qquad |\!\leftarrow\! 5 \!\rightarrow\!|$$

在数学模式下也可以直接使用文本模式的水平间距命令（参见 2.1.5.1 节），不过通常使用的只有 \quad 和 \qquad，常用来分隔公式中的不同部分，如：

4-5-15

```
\[
  f(x) \equiv 0, \qquad x > 0
\]
```

$$f(x) \equiv 0, \qquad x > 0$$

技巧性地使用负间距是用来拼合新的数学符号的方法之一。例如，在各种数学字体中都没有上积分和下积分的符号，就可以通过 \mspace 反复平移普通积分号和上下画线得到上下积分的符号。我们以下积分为例，第一个间距控制下画线的位置，第二个间距控制下画线的长度，第三个间距是前两个值之和的相反数退回原点，把普通的 \int 命令放在最后可以保证积分号的上下标不会走样，这里具体间距的数值可能需要根据具体使用的数学字体进行调整。

4-5-16

```
\newcommand\lowint{%
  \mspace{2mu}\underline{\vphantom{\int}\mspace{7mu}}\mspace{-9mu}\int}
\[
  \lowint_a^b f(x)\,\mathrm{d}x = \inf_P s(P).
\]
```

$$\underline{\int}_a^b f(x)\,\mathrm{d}x = \inf_P s(P).$$

LaTeX 中还有许多与数学公式相关的长度变量，必要时可以使用 \setlength 等命令进行相关的设置，这里是一些常用的变量：

- 文档类使用 fleqn 选项时，显示公式会以一个固定缩进左对齐排版（参见 2.4.1 节、4.1 节）。这个固定的缩进距离由 \mathindent 控制。

- \abovedisplayshortskip 和 \abovedisplayskip 控制显示公式与上面文字内容的距离。区别如果前面的文字行在公式开始之前就结束，即公式和前面的文字没有重叠部分时，使用 \abovedisplayshortskip，否则使用 \abovedisplayskip，它们的默认值是：

```
\abovedisplayskip=12pt plus 3pt minus 9pt
\abovedisplayshortskip=0pt plus 3pt
```

如果公式和文字没有重叠，间距要比有重叠时小得多，这样可以保证视觉上的平衡。默认值原本是对 10pt 左右的 Computer Modern 字体设置的，当正文使用特别大或特别小的字号时，就有必要对上面的默认值做适当的修改了（注意 \large、\tiny 等字号命令也不修改这些参数，参见 2.1.4 节），例如：

```
\zihao{7}% 5.5bp
\setlength{\abovedisplayskip}%
  {2pt plus 1pt minus 3pt}
当文字非常小时也应该同时减小
显示公式与文字的间距：
\[ 1+2+3+4+5 = 15 \]
```

当文字非常小时也应该同时减小显示公式与文字的间距：

$$1+2+3+4+5=15$$

4-5-17

- \belowdisplayshortskip 和 \belowdisplayskip 控制显示公式与下面文字内容的距离，用法与控制上间距的变量 \abovedisplayshortskip, \abovedisplayskip 类似，其默认值为：

```
\belowdisplayskip=12pt plus 3pt minus 9pt
\belowdisplayshortskip=7pt plus 3pt minus 4pt
```

- 变量 \jot 用来控制多行公式之间的间距，默认值是 3pt。有些公式可以设置它来调整过于紧密的间距，同样在大幅度改变字号时可能需要同时修改 \jot 的值才能得到正确的公式，例如：

```
\setlength\jot{9pt}% 用来分开分式
\begin{gather}
a = \frac12 \\ b = \frac34
\end{gather}
```

$$a=\frac{1}{2} \tag{4.33}$$

$$b=\frac{3}{4} \tag{4.34}$$

4-5-18

当然，如果只是调整个别公式，设置 \jot 值的方式可能不如直接用 \\[6pt] 这种用法直接灵活，因此最好全局地使用这类设置。

- \thinmuskip 是细数学间距，它将影响 \, 和 \! 和命令的间距值，默认值是 3mu。
- \medmuskip 是中数学间距，它将影响 \: 或 \> 命令的间距值，默认值是 4mu plus 2mu minus 4mu。
- \thickmuskip 是厚数学间距，它将影响 \; 命令的间距值，默认值是 5mu plus 5mu。
- \mathsurround 控制行内数学公式与两边文字的额外间距，默认值为 0pt。注意这个命令控制的是额外间距，一般数学公式与文字之间仍然应该使用空格分开，例如：

4-5-19

```
\setlength{\mathsurround}{3pt}
公式 $a+b$ 与文字比较松散。
```

公式 $a+b$ 与文字比较松散。

对中文它是多余的，xeCJK 宏包会在数学公式与汉字之间添加间距，并且由于它会使诸如“p-adic 数”这样的词汇产生难看的间距，所以使用时应该慎重。

\skew 命令可以用来调整数学重音的位置。这个命令带有三个参数，第一个参数是重音符号要移动的距离（一个数字，单位是 mu），后面两个参数分别是数学重音的位置和被标记的数学符号，例如把双点左移到字母 h 的一竖正上方：

4-5-20

```
$\ddot{h} \iff \skew{-2}{\ddot}{h}$
```

$\ddot{h} \iff \skew{-2}{\ddot}{h}$

2.1.1.3 节见到的幻影（phantom）也是在数学公式中常用的一种间距。\vphantom 可以作为支架使用，而 \hphantom 和 \phantom 则常用来产生特殊的对齐效果：

4-5-21

```
\begin{equation*}
\begin{split}
f(x) &= \left(\vphantom{\frac1x}
x+2+3+4\right. \\
  & \left.\phantom{=\biggl(x+{}}
5+6+7+\frac1x \right)^2 \\
  &= g(x)
\end{split}
\end{equation*}
```

$$
\begin{aligned}
f(x) &= \left(\vphantom{\frac1x}
x+2+3+4\right. \\
&\quad \left.\phantom{=\biggl(x+{}}
5+6+7+\frac1x \right)^2 \\
&= g(x)
\end{aligned}
$$

最后我们来看 \smash 命令。\smash 的功能与 \vphantom 正相反，它显示参数中的内容，但好像内容并没有高度和深度一样，如：

```
\[  \underline{
\smash{\int f(x)\,\mathrm{d}x}
}  \]
```

$$\underline{\smash{\int f(x)\,\mathrm{d}x}}$$

4-5-22

amsmath 给 \smash 增加了 t、b 两个可选参数，分别表示只忽略内容盒子的高度和深度，这让 \smash 命令变得更加有用，如调整根号的高度：

```
$\sqrt{A_{n_k}} \qquad
\sqrt{\smash[b]{A_{n_k}}}$
```

$\sqrt{A_{n_k}}$　　$\sqrt{\smash[b]{A_{n_k}}}$

4-5-23

或是用来得到这样的括号：

```
\vspace{\baselineskip}% 被忽略的高度
\[
  \text{实数} \begin{cases}
    \text{有理数}\smash[t]{\begin{cases}
      \text{整数}\smash{\begin{cases}
          \text{奇数} \\ \text{偶数}
        \end{cases}}\\
      \text{分数}
    \end{cases}} \\[4ex]
    \text{无理数}\smash[b]{\begin{cases}
      \text{代数无理数} \\ \text{超越数}
    \end{cases}}
  \end{cases}
\]
```

$$\text{实数}\begin{cases}\text{有理数}\begin{cases}\text{整数}\begin{cases}\text{奇数}\\ \text{偶数}\end{cases}\\ \text{分数}\end{cases}\\[4ex] \text{无理数}\begin{cases}\text{代数无理数}\\ \text{超越数}\end{cases}\end{cases}$$

4-5-24

本章注记

有关 LaTeX 公式排版的文档非常丰富，所有 LaTeX 的基本书籍和文档都会把它作为重点内容。 Voß [281] 是 LaTeX 数学模式的一个相当全面的手册，里面也有大量的工具说明和技巧性的示范； American Mathematical Society [7] 和 Mittelbach and Goossens [166, Chapter 8] 给出了 $\mathcal{A}\mathcal{M}\mathcal{S}$-LaTeX 全面介绍，后者也包括大量技巧和示例。

全面的数学符号的参考手册仍属 Pakin [192]，而选用数学字体的绝佳参考则见于 Hartke [100] 和 Jørgensen [118]。数学字体的选用及其与正文字体的交互参见 2.1.3 节“字体”。

Knuth [126, Chapter 16–19, 26], Mittelbach and Goossens [166, Chapter 8] 和 盖鹤麟（Gai, Helin） [313] 既是数学公式方面的完整参考，也提供了有关数学公式精调的许多原则和手段，特别是一些美学上的准则。 ISO/TC 12 [114] 则提出了 ISO 组织对科技文档数量及单位使用的一些要求。

数学模式中的 `array` 环境可以提供向量和矩阵的细节控制，参见 5.1.1 节“`tabular` 和 `array`”；数学交换图表的绘制参见 5.5.1 节“Xy-pic 与交换图表”。

第5章

绘制图表

图表的制作大概是LaTeX中最令人着迷的部分了，为图表编写的宏包、工具、书籍、文档数不胜数，LaTeX在这方面所能达到的效果也从最简单的直线稿图、简单表格发展到极为复杂的图形图像、数据报表，其功能不亚于许多专业图表软件。但另一方面，缺乏直观的代码也让不少人将其视为畏途。让图表问题变得更容易，是许多LaTeX用户的愿望。这一章我们就要进入这个全新的领域，从基本的工具开始，渐次发散开来，逐步领略个中妙趣。

注意，本章出现在示例和习题中的多数表格格式与全书其他部分可能有较大区别，展示的是示例代码产生的未经调整的格式。除5.3.2节外，图表标题格式仍与全书统一。

5.1 LaTeX中的表格

在语义上，表格的作用是展示多种相关的内容，而在形式上，表格就是按行和列对齐的一组内容。表格是二维延伸的特殊排版对象，与tabbing环境简单地预设对齐位置不同，在表格中较后面内容的宽度也会影响前面内容的排列。在LaTeX中，表格是逐行输入的，可以设置表格的列对齐格式和表格线，通过扩展的宏包还可以达成一些特殊的效果。

5.1.1 tabular和array

在LaTeX中，表格使用两个环境录入：在文本或数学模式下都可以使用tabular环境，在数学模式下还可以使用array环境。在数学模式下使用tabular环境，其表项

内容也是按文本模式排版的。除了所在的模式不同，tabular 环境和 array 环境在功能使用上基本没有区别。一般使用 tabular 环境排版表格，而用 array 环境排版包含数学符号的公式，如复杂矩阵等。

tabular 与 array 环境的一般格式为：

```
\begin{tabular}[⟨垂直对齐⟩]{⟨列格式说明⟩}
⟨表项⟩ & ⟨表项⟩ & ...& ⟨表项⟩ \\
.....
\end{tabular}
\begin{array}[⟨垂直对齐⟩]{⟨列格式说明⟩}
⟨表项⟩ & ⟨表项⟩ & ...& ⟨表项⟩ \\
.....
\end{array}
```

此外环境中还可能有表格线等命令。

tabular 与 array 环境中，每行后用 \\ 表示换行．一行之内的不同列之间用 & 分开，用 & 和 \\ 就可把表格分成许多个表项。

表格的可选垂直对齐选项很少使用，常用的列格式说明就是左、中、右对齐，一个简单表格的例子如：

5-1-1

```
\begin{tabular}{lcr}
left & center & right \\
本列左对齐 & 本列居中
  & 本列右对齐 \\
\end{tabular}
```

left	center	right
本列左对齐	本列居中	本列右对齐

表格中的一个表项隐含一个分组，因此在表项内部的声明命令，其作用域也以 & 和 \\ 为界，如：

5-1-2

```
\begin{tabular}{ll}
\bfseries 功能 & \bfseries 环境 \\
表格 & \ttfamily tabular \\
对齐 & \ttfamily tabbing \\
\end{tabular}
```

功能	**环境**
表格	tabular
对齐	tabbing

在列格式说明中可以用 | 表示画一条竖线，而在表格一行前后使用 \hline 命令可以画一条横线，如：

```
\[
  \begin{array}{r|r}
    \frac12 & 0 \\
    \hline
    0 & -\frac12 \\
  \end{array}
\]
```

$$\begin{array}{r|r} \frac{1}{2} & 0 \\ \hline 0 & -\frac{1}{2} \end{array}$$

5-1-3

表格中，可选的 ⟨垂直对齐⟩ 参数可以是：

- **t**　按表格顶部对齐，顶部是表格第一行或表线。
- **b**　按表格底部对齐，底部是表格最后一行或表线。
- **默认**　垂直居中，非 t 和 b 的参数都看做是居中。

例如：

```
\begin{tabular}[b]{c}
上 \\ 中间 \\ 下
\end{tabular}
与底部对齐。
```

上
中间
下　与底部对齐。

5-1-4

tabular 环境和 array 环境得到的表格都只是一个普通的盒子，因此表格与文字或数学公式的其他部分通常会直接连在一起，例如：

```
\begin{tabular}{|rr|}
\hline
  输入 & 输出 \\ \hline
  $-2$ & 4 \\
  0 & 0 \\
  2 & 4 \\
\hline
\end{tabular}
\qquad
输入与输出有关系 $y = x^2$
```

输入	输出
−2	4
0	0
2	4

输入与输出有关系 $y = x^2$

5-1-5

不过多数表格通常并不在前后有文字，因此可以放在专门的环境中。文档中的表格经常被放在带有编号、标题的浮动体中，这样可以保证表格与前后文字不直接相连，也能避免难看的分页，参见 5.3.1 节。

完整的列格式说明符如下：

- `l` 本列左对齐。
- `c` 本列居中。
- `r` 本列右对齐。
- `p{⟨宽⟩}` 本列具有固定宽度，且可以自动换行。事实上，这相当于表格项是由命令 `\parbox[t]{⟨宽⟩}` 得到的，但为了避免混淆，里面不能直接使用 `\\` 命令。如果需要不同的对齐方式，可以在表项中直接加 `\centering`, `\raggedleft` 等命令。
- `|` 画一条竖线，不占表项计数。
- `@{⟨内容⟩}` 添加任意内容，不占表项计数。使用 `@{⟨内容⟩}` 列格式符会同时取消表列间的距离。
- `*{⟨计数⟩}{⟨列格式说明⟩}` 作为一种简写，将给出的列格式说明符重复多次。

使用这些说明符就可以得到相当多变的表列格式，常见的普通表格都不难获得，例如：

```
\begin{tabular}{|c|rrr|p{4em}|}
\hline
  姓名 & 语文 & 数学 & 外语 & 备注 \\
\hline
  张三 & 87 & 100 & 93 & 优秀 \\
  李四 & 75 & 63 & \emph{52} & 补考另行通知 \\
  王小二 & 80 & 82 & 78 & \\
\hline
\end{tabular}
```

5-1-6

姓名	语文	数学	外语	备注
张三	87	100	93	优秀
李四	75	63	*52*	补考另行通知
王小二	80	82	78	

又如，使用 `@` 格式符插入小数点，在左右两边分别使用右对齐和左对齐，就可以得到按小数点对齐的表格：

```
\begin{tabular}{|c|r@{.}l|}
  \hline
  收入 & 12345&6  \\ \hline
  支出 &   765&43 \\ \hline
  节余 & 11580&17 \\ \hline
\end{tabular}
```

收入	12345.6
支出	765.43
节余	11580.17

5-1-7

而且可以用 * 格式重复这一模式，输入多列数据（这里用数学模式方便负数排版）：

```
\[
\begin{array}{|c|*{3}{r@{.}l|}}  % 相当于 |c|r@{.}l|r@{.}l|r@{.}l|
\hline
\text{收入} & 12345&6  & 5000&0 &   1020&55 \\ \hline
\text{支出} &   765&43 & 5120&5 &  98760&0  \\ \hline
\text{节余} & 11580&17 & -120&5 & -97739&45 \\ \hline
\end{array}
\]
```

5-1-8

收入	12345.6	5000.0	1020.55
支出	765.43	5120.5	98760.0
节余	11580.17	−120.5	−97739.45

不过按小数点对齐这一功能也可以用 dcolumn 宏包[46]更清晰也更自然地处理，它提供了一种新的列格式说明符 D，这个说明符带有三个参数，分别表示小数点的输入方式、小数点的输出方式和最大小数位数（负值表示不限）。为了正常使用负号等数学符号，使用 D 说明符的列将自动进入数学模式，因此对文字表头要做特别的处理，通常是使用 \multicolumn 命令（参见 5.1.2 节）改变表头的列格式。dcolumn 宏包会自动调用 array 宏包，为使用方便，可以用 array 宏包的 \newcolumntype 功能再将带预设参数的 D 说明符定义为新的列说明符（参见 5.1.6 节），例如：

```
% 导言区 \usepackage{dcolumn}
\newcolumntype{d}{D{.}{.}{2}}
\begin{tabular}{|c|*{3}{d|}}  % 相当于 |c|d|d|d|
\hline
姓名 & \multicolumn{1}{c|}{张三} & \multicolumn{1}{c|}{李四}
```

```
    & \multicolumn{1}{c|}{王五} \\ \hline
收入 & 12345.6  & 5000   &   1020.55 \\ \hline
支出 &   765.43 & 5120.5 &  98760    \\ \hline
节余 & 11580.17 & -120.5 & -97739.45 \\ \hline
\end{tabular}
```

5-1-9

姓名	张三	李四	王五
收入	12345.6	5000	1020.55
支出	765.43	5120.5	98760
节余	11580.17	−120.5	−97739.45

表格中，列与列之间最小距离的一半是由变量 \tabcolsep 和 \arraycolsep 控制的，可以单独进行设置。在第一列之前和最后一列的后面也同样有宽度为 \tabcolsep 或 \arraycolsep 的间距，不过可以使用 @{} 说明符去除，例如：

```
\verb=tabular= 环境可以在
$\left(\begin{tabular}{@{}c@{}}
  文本 \\ 数学
\end{tabular}\right)$
模式下通用。
```

5-1-10

tabular 环境可以在 $\left(\begin{tabular}{@{}c@{}}文本\\数学\end{tabular}\right)$ 模式下通用。

如果在表列的 @ 说明符中使用 \extracolsep{⟨间距⟩}，可以增加它之后的所有表列左侧的额外间距，但注意 @ 说明符本身会消除原有的列间距，例如：

```
% 第 1 列前是原始间距，第 2 列前只有 1em 间距
% 第 3、4 列前则是原始间距加 1em
\begin{tabular}{|c|@{\extracolsep{1em}}c|c|c|}
\hline
  1 & 2 & 3 & 4 \\
  1 & 2 & 3 & 4 \\
\hline
\end{tabular}
```

5-1-11

1	2	3	4
1	2	3	4

表格中行与行之间的距离可以由宏 \arraystretch 控制。\arraystretch 默认定义为 1，可以通常重定义得到指定倍数的表格行距，例如：

```
\renewcommand\arraystretch{2}
\begin{tabular}{|l|r|}
\hline
这是一个 & 宽松的表格 \\ \hline
loose & table \\ \hline
\end{tabular}
```

这是一个	宽松的表格
loose	table

5-1-12

使用表格的垂直对齐选项 t 与 b 时，如果同时顶部或底部有表格线，则前后的文字会与表格线而不是首末的表行对齐。表格的这个性质在很多时候并不符合我们的需求，此时可以使用 array 宏包[163] 提供的 \firsthline 和 \lasthline 命令代替 \hline，例如：

```
% 导言区 \usepackage{array}
\begin{tabular}[b]{|c|}
\firsthline
上 \\ 中间 \\ 下 \\
\lasthline
\end{tabular}
与底部对齐。
```

上 中间 下

与底部对齐。

5-1-13

练 习

5.1 排版下面的增广矩阵：

$$\left(\begin{array}{ccc|c} a_{11} & a_{12} & a_{13} & b_1 \\ a_{21} & a_{22} & a_{23} & b_2 \\ a_{31} & a_{32} & a_{33} & b_3 \end{array}\right)$$

5.2 排版下面的表格，并考虑，怎样的列对齐方式更为合理，表格线该如何设置更为美观。

人数	患慢性支气管炎	未患慢性支气管炎	合计
吸烟	43	162	205
不吸烟	13	121	134
合计	56	283	339

5.3 查阅 siunitx 宏包的文档 Wright [298]，了解 siunitx 提供的 S 列格式的用法，并排版表 5.1。

表 5.1 使用 siunitx 排版的数量表

数量
−2 147 483 648
3.141 592 65
$2.997\,924\,58 \times 10^{8}$
3.55 ± 0.02

5.1.2 表格单元的合并与分割

`\multicolumn{⟨项数⟩}{⟨新列格式⟩}{⟨内容⟩}` 命令可用于将一行中几个不同的表项合并为一项，它经常用于排版跨列的表头，例如：

5-1-14

```
\begin{tabular}{|r|r|}
  \hline
  \multicolumn{2}{|c|}{成绩} \\ \hline
  语文 & 数学 \\ \hline
  87 & 100 \\ \hline
\end{tabular}
```

成绩	
语文	数学
87	100

注意这里合并的新列格式里面只能有一个 `c`、`l`、`r` 或 `p{⟨宽⟩}`，以及可选的 `@` 选项和表线。`\multicolumn` 会重定义它所产生的列后面的竖线（如果是第一列，也包括前面的竖线），当表格有竖线时，要特别注意不要让 `\multicolumn` 增加或减少应有的竖线，当然，也可以用它来产生间断的竖线。`\multicolumn` 命令不仅可以用于合并多列，也可以只“合并”一列，作用是改变所在表项的对齐、竖线格式，例如：

5-1-15

```
\begin{tabular}{|r|r|}
  \hline
  \multicolumn{1}{|c|}{输入} &
  \multicolumn{1}{c|}{输出} \\ \hline
  1 & 1 \\ 5 & 25 \\ 15 & 225 \\ \hline
\end{tabular}
```

输入	输出
1	1
5	25
15	225

\cline 命令与 \hline 命令类似，都用来画水平的表格线。不过 \cline 带有一个形如 ⟨起⟩-⟨止⟩ 参数，用来说明表格线起始和终止的列号，用来画出不完全或间断的横线。我们来扩充前面的例 5-1-14：

```
\begin{tabular}{|c|r|r|}
\hline
& \multicolumn{2}{c|}{成绩
} \\ \cline{2-3}
姓名 & 语文 & 数学 \\ \hline
张三 & 87 & 100 \\ \hline
\end{tabular}
```

	成绩	
姓名	语文	数学
张三	87	100

5-1-16

从上面的例子可以看出，使用 \cline 命令画出一段不完全的表线，就可以产生跨行表格项的效果。

与 \multicolumn 命令相反，\vline 命令可以在表项内部画一条只占一行高度的竖线。如果用它来拆分已有的表项，要注意加上适当的间距[①]，例如：

```
\begin{tabular}{|c|}
\hline
1 \\ \hline
1 \vline\ 2 \\ \hline % 加一个空格的间距
1 \vline\ 2 \vline\ 3 \\ \hline
\end{tabular}
```

1
1 | 2
1 | 2 | 3

不过，使用 \vline 拆分表项不易掌握间距，另一种方式是直接使用嵌套的表格，此时应该注意在两侧只使用 @{}，避免间距和竖线，例如：

```
\begin{tabular}{|c|}
\hline
1 \\ \hline
\begin{tabular}{@{}c|c@{}} 1 & 2 \end{tabular} \\ \hline
\begin{tabular}{@{}c|c|c@{}} 1 & 2 & 3\end{tabular} \\ \hline
\end{tabular}
```

5-1-17

① 准确的间距是 \tabcolsep。

1		
1	2	
1	2	3

直接使用 \cline 画线模拟的跨行表项在行数是偶数时并不正确，通常跨行的新表项应该在两行的正中间，而不是在上面或下面一行。为此，可以使用 multirow 宏包[191]提供的 \multirow 命令排版跨行的表项。\multirow 命令的基本语法格式如下：

\multirow{⟨行数⟩}{⟨宽度⟩}{⟨内容⟩}
\multirow{⟨行数⟩}*{⟨内容⟩}

使用前一种形式，内容达到宽度后会自动换行；使用后面一种形式，产生表项的宽度就是输入内容的宽度。通常后一种形式更常用些，沿用前面的例 5-1-16：

5-1-18

```
% 导言区 \usepackage{multirow}
\begin{tabular}{|c|r|r|}
\hline
\multirow{2}*{姓名} &
\multicolumn{2}{c|}{成绩} \\ \cline{2-3}
     & 语文 & 数学 \\ \hline
张三 & 87   & 100  \\ \hline
\end{tabular}
```

姓名	成绩	
	语文	数学
张三	87	100

makecell 宏包[141]提供的 \makecell 命令可以单独控制表项单元，可以在表项中使用 \\ 命令自由地换行。在不打算固定表列宽度时，它比 p{⟨宽度⟩} 选项更为灵活，如：

5-1-19

```
% 导言区 \usepackage{makecell}
\begin{tabular}{|r|r|} \hline
\makecell{处理前\\数据} &
\makecell{处理后\\数据} \\ \hline
4934 & 8945 \\ \hline
\end{tabular}
```

处理前 数据	处理后 数据
4934	8945

*本节剩余内容初次阅读可略过。

\makecell 命令的内容默认使用居中对齐，也可以使用可选选项 t、b、l、r、c 等分别控制其垂直与水平对齐方式为顶部、底部、左对齐、右对齐或居中。

makecell 宏包的这种表项分行常用在表头中。事实上，它还为表头单独定义了与 \makecell 类似的 \thead 命令，它产生字体较小、上下间距较大的单元，更适合文字较多的多行表头使用，例如：

```
% 导言区 \usepackage{makecell}
\begin{tabular}{|r|r|}
\hline
\thead{处理前\\数据} &
\thead{处理后\\数据} \\ \hline
4934 & 8945 \\
\hline
\end{tabular}
```

处理前 数据	处理后 数据
4934	8945

5-1-20

makecell 的 \rothead 命令则相当于旋转了 90° 的 \thead 命令，这个命令还依赖 rotating 宏包[72]。使用 \rothead 时需要给旋转表头的宽度 \rotheadsize 赋值，例如：

```
% 导言区 \usepackage{rotating,makecell}
\settowidth\rotheadsize{\theadfont 数学课}
\begin{tabular}{|c|c|}
\hline
\thead{姓名} & \rothead{数学课\\成绩} \\\hline
张三 & 100 \\\hline
\end{tabular}
```

姓名	数学课 成绩
张三	100

5-1-21

表头的字体由 \theadfont 命令控制，默认是 \foonotesize；间距由 \theadgape 或 \rotheadgape 生成，默认是 \gape；对齐是由 \theadalign 控制，默认是 {cc}。可以重定义这些命令来控制表头的格式，详细内容可参见 makecell 宏包的文档。

makecell 还提供了 \gape{⟨内容⟩} 命令，用来增加表项内容的上下间距，默认是增加 \jot 的距离，可选的 t、b 参数可以只改变表项顶部或底部的间距。也可以使用 \Gape[⟨间距⟩]{⟨内容⟩} 或 \Gape[⟨上间距⟩][⟨下间距⟩]{⟨内容⟩} 手工设置增加的间距。

no gape	\gape[t]	\gape[b]	\gape	\Gape[6pt]

使用 \setcellgapes{⟨间距⟩} 可以设置表项内容的竖直间距，然后可以使用 \makecellgapes 和 \nomakecellgapes 命令对表格中的所有表项打开或关闭这个间距功能。

如果同时使用 multirow 宏包和 makecell 宏包，命令 \multirowcell 和 \multirowthead 命令则成为 \makecell、\thead 与 \multirow 的结合体，可以在跨行的表项中随意地使用 \\ 命令换行，例如：

5-1-22

```
% 导言区 \usepackage{multirow,makecell}
\begin{tabular}{|c|r|}
\hline
\multirowcell{3}{各科\\成绩} & 78 \\
\cline{2-2} & 82 \\ \cline{2-2}
 & 86 \\ \hline
\end{tabular}
```

各科 成绩	78
	82
	86

作者编写的 diagbox 宏包[306] 提供了 \diagbox 命令，可以用来对表头进行斜线分割，其基本语法格式如下：

\diagbox[⟨选项⟩]{⟨左⟩}{⟨右⟩}
\diagbox[⟨选项⟩]{⟨左⟩}{⟨中⟩}{⟨右⟩}

一般情况下 ⟨选项⟩ 可以省略，例如：

5-1-23

```
% 导言区 \usepackage{diagbox}
\begin{tabular}{|c|*{4}{c}|}
\hline
\diagbox{天干}{地支} & 子 & 丑 & 寅 & 卯 \\
\hline
甲 &  1 && 51 & \\
乙 &&  2 && 52 \\
丙 & 13 && 3 & \\
丁 && 14 && 4\\
\hline
\end{tabular}
```

地支 / 天干	子	丑	寅	卯
甲	1		51	
乙		2		52
丙	13		3	
丁		14		4

\diagbox 自动判断是把表头分成两部分还是三部分，例如：

```
% 导言区 \usepackage{diagbox}
\begin{tabular}{|c|*{4}{c}|}
\hline
\diagbox{天干}{序号}{地支} & 子 & 丑 & 寅 & 卯 \\
\hline
甲 &  1 && 51 & \\
乙 &&  2 && 52 \\
丙 & 13 && 3 & \\
丁 && 14 && 4\\
\hline
\end{tabular}
```

5-1-24

序号 / 地支 / 天干	子	丑	寅	卯
甲	1		51	
乙		2		52
丙	13		3	
丁		14		4

\diagbox 的 ⟨选项⟩ 可以使用 width 和 height 设置斜线表头的高度和宽度，用 dir 选项设置斜线表头的方向，或用 trim 选项设置左右边界等，详细用法可参见宏包手册 [306]。

diagbox 宏包的前身是 slashbox 宏包，它提供了 \slashbox 和 \backslashbox 命令，二者都带有两个内容参数，可以把一个表项用斜线拆分开来，其中 \backslashbox 命令与带有 ⟨左⟩、⟨右⟩ 两个参数的 \diagbox 命令相似。作为 slashbox 的代替品，diagbox 宏包提供了对 slashbox 宏包的向后兼容性，也可以在 diagbox 中使用 \slashbox 与

\backslashbox 命令。

除了 diagbox，makecell 宏包也提供了一个 \diaghead 命令生成斜线表头，不过它语法较为冗长，效果也不如 \diagbox，这里就不做介绍了。

练 习

5.4 绘制如下梯形表格：

张家村	三里		
李家屯	六里	三里	
王家庄	五里	八里	七里
	赵镇	张家村	李家屯

5.5 makecell 宏包的 \makecell 实际上是实现了一个只有一列的小表格，tabular 环境可以嵌套，请据此给出一个自己的 \makecell 命令的定义方法。

5.6 绘制如下表格：

摩擦副配对材料	锁紧结构 \ 锁紧力矩 N·m		供油压力（MPa）				
			2	5	7	10	12
调制钢	摩擦锥	$\theta = 60°$，单槽	1.3	2.6	2.9	3.5	3.8
		$\theta = 60°$，双槽	1.8	3.4	3.7	4.5	5.2
		$\theta = 64°$，双槽	1.2	2.6	2.9	3.4	3.7
		$\theta = 36°$，双槽	1.5	3.3	4.3	4.4	4.6
		$\theta = 20°$，双槽	1.6	3.3	3.8	5.0	5.5
H62	内摩擦环		1.6	2.9	4.0	4.6	5.1
	外摩擦环		7.6	8.6	9.5	10.5	11.7

5.1.3 定宽表格与 tabularx

tabular 环境得到的表格宽度就是各列自然宽度的总和，可以使用 p{⟨宽⟩} 格式规定单列的宽度，但有时还需要规定整个表格的总宽度，比如让表格铺满整页。

LaTeX 的 tabular* 环境就是一种固定宽度的表格，相较于普通的 tabular 环境，它增加了一个 ⟨宽度⟩ 参数：

*本节内容初次阅读可略过。

\begin{tabular*}{⟨宽度⟩}[⟨垂直对齐⟩]{⟨列格式说明⟩}
**⟨表项⟩ & ⟨表项⟩ & ...& ⟨表项⟩ **
.....
\end{tabular*}

要让 tabular* 环境发挥作用，一般需要使用 \extracolsep 命令给表列增加弹性宽度，例如：

```
\begin{tabular*}{\textwidth}{|c@{\extracolsep{\fill}}ccccc|}
\hline
  数字 & 1 & 2 & 3 & 4 & 5 \\
  字母 & A & B & C & D & E \\
  天干 & 甲 & 乙 & 丙 & 丁 & 戊 \\
\hline
\end{tabular*}
```

5-1-25

数字	1	2	3	4	5
字母	A	B	C	D	E
天干	甲	乙	丙	丁	戊

tabular* 环境使用不便，在画竖直表线和设置列间距的时候都很容易出问题。为了简化定宽表格的排版，可以使用 tabularx 宏包[45] 提供的 tabularx 环境。tabularx 环境的语法与 tabular* 相同，只是它提供了一个 X 列格式说明符，表示自动延伸的表列，可以与其他列说明符一起使用。X 列格式很好地处理了间距问题，表项内容会按一个定宽的 \parbox 盒子排版，可以自动换行，内部默认是左对齐的，例如：

```
% 导言区 \usepackage{tabularx}
\begin{tabularx}{\textwidth}{|c|X|X|X|X|X|}
\hline
  数字 & 1 & 2 & 3 & 4 & 5 \\ \hline
  字母 & A & B & C & D & E \\ \hline
  天干 & 甲 & 乙 & 丙 & 丁 & 戊 \\
\hline
\end{tabularx}
```

5-1-26

数字	1	2	3	4	5
字母	A	B	C	D	E
天干	甲	乙	丙	丁	戊

tabularx 宏包依赖 array 宏包，也可以使用 array 宏包的机制（参见 5.1.6 节）设定其他的对齐方式。由于 \centering 等命令会影响 \\ 命令的定义，需要在对齐命令后面加上 \arraybackslash 命令恢复[①]。例如，可以用 \newcolumntype 定义一个居中的 Y 列格式：

```
% 导言区 \usepackage{tabularx}
\newcolumntype{Y}{>{\centering\arraybackslash}X}
\begin{tabularx}{\textwidth}{|c|Y|Y|Y|Y|Y|}
\hline
  数字 & 1 & 2 & 3 & 4 & 5 \\ \hline
  字母 & A & B & C & D & E \\ \hline
  天干 & 甲 & 乙 & 丙 & 丁 & 戊 \\
\hline
\end{tabularx}
```

5-1-27

数字	1	2	3	4	5
字母	A	B	C	D	E
天干	甲	乙	丙	丁	戊

5.1.4 长表格与 longtable

非常长的表格也很常见，特别是格式整齐的数据报表。由于表格本身是一个不能断开的盒子，直接在正文中使用大表格一般会造成难看的分页。因此，较大的表格一般放在浮动体 table 环境中（参见 5.3.1 节）。没有标题、只有两三行的短表格才适合放在正文中，不过通常还是放在 center、quote 等环境中，不与段落中的文字直接相连。

只要表格的内容不超过一页，使用浮动体总是比较合适的方式。但是，如果表格内容已经超出一页的范围，就必须对表格进行拆分才行。longtable 就是处理这种行数非常多的长表格的宏包[49]。

longtable 宏包提供一个 longtable 环境，它的语法格式与 tabular 环境类似，不过增加了一些处理跨页时表格头尾的新命令：

① 不定义新的列格式的话，可以只在最后一列添加。

*本节后面内容初次阅读可略过。

```
\begin{longtable}[⟨水平对齐⟩]{⟨列格式说明⟩}
⟨表头⟩
\endhead
⟨第一页表头⟩
\endfirsthead
⟨表尾⟩
\endfoot
⟨最后一页表尾⟩
\endlastfoot
⟨表项⟩ & ⟨表项⟩ & ...& ⟨表项⟩ \\
.....
\end{longtable}
```

基本的 longtable 用法就是简单地使用 longtable 环境，不使用 \endhead 等任何特别的命令，表格会在分页时自动断开。⟨水平对齐⟩选项可以选择左对齐 l、右对齐 r 或居中 c，默认设置是居中。\endhead、\endfirsthead、\endfoot、\endlastfoot 几个命令则定义了表格在换页时的行为，它们用在 longtable 环境的最前面，把 longtable 环境的前面一部分划分为四个（也可以省略）部分，分别表示表格在每一页和第一页的表头、每一页和最后一页的表尾。与 tabular 环境不同，longtable 环境中可以使用脚注、表格标题（参见 5.3.1 节）等命令。表格的标题一般放在第一页表头或最后一页表尾的地方，即 \endfirsthead 和 \endlastfoot 命令所隔开的部分，而用 \endhead 和 \endfoot 隔开的部分则一般用来填写表格续页的说明。

例如，longtable 宏包的常用选项和命令汇总，就可以用下面的长表格得到：

```
% 导言区使用 \usepackage{longtable}
\newcommand\meta[1]{\emph{$\langle$#1$\rangle$}}
\begin{longtable}{|l|l|}
\caption{\texttt{longtable} 环境中的命令汇总} \\
\hline
\endfirsthead
\multicolumn{2}{l}{（续表）} \\
\hline
\endhead
\hline
\multicolumn{2}{c}{\itshape 接下一页表格……} \\[2ex]
```

```
\endfoot
\hline
\endlastfoot
\multicolumn{2}{|c|}{环境的水平对齐可选项} \\ \hline
留空 & 表格居中%
\footnote{实际上，留空的对齐方式是由一组命令控制的，参见宏包文档。} \\
\verb=[c]= & 表格居中 \\
\verb=[l]= & 表格左对齐 \\
\verb=[r]= & 表格右对齐 \\
\hline \multicolumn{2}{|c|}{结束表格一行的命令} \\ \hline
\verb=\\= & 普通的结束一行表格 \\
\verb=\\[=\meta{距离}\verb=]= & 结束一行，并增加额外间距 \\
\verb=\\*= & 结束一行，禁止在此分页 \\
\verb=\kill= & 当前行不输出，只参与宽度计算 \\
\verb=\endhead= & 此命令以上部分是每页的表头 \\
\verb=\endfirsthead= & 此命令以上部分是表格第一页的表头 \\
\verb=\endfoot= & 此命令以上部分是每页的表尾 \\
\verb=\endlastfoot= & 此命令以上部分是表格最后一页的表尾 \\
\hline \multicolumn{2}{|c|}{标题命令} \\ \hline
\verb=\caption{=\meta{标题}\verb=}= & 生成带编号的表格标题 \\
\verb=\caption*{=\meta{标题}\verb=}= & 生成不带编号的表格标题 \\
\hline \multicolumn{2}{|c|}{分页控制} \\ \hline
\verb=\newpage= & 强制分页 \\
\verb=\pagebreak[=\meta{程度}\verb=]= & 允许分页的程度（0--4） \\
\verb=\nopagebreak[=\meta{程度}\verb=]= & 禁止分页的程度（0--4） \\
\hline \multicolumn{2}{|c|}{脚注控制} \\ \hline
\verb=\footnote= & 使用脚注\footnote{普通表格中不能用。},
  注意不能用在表格头尾 \\
\verb=\footnotemark= & 单独产生脚注编号，不能用在表格头尾 \\
\verb=\footnotetext= & 单独产生脚注文字 \\
\hline \multicolumn{2}{|c|}{长度参数} \\ \hline
\verb=\LTleft= & 对齐方式留空时，表格左边的间距，默认为 \verb=\fill= \\
\verb=\LTright= & 对齐方式留空时，表格右边的间距，默认为 \verb=\fill= \\
```

```
\verb=\LTpre= & 表格上方间距，默认为 \verb=\bigskipamount= \\
\verb=\LTpost= & 表格下方间距，默认为 \verb=\bigskipamount= \\
\verb=\LTcapwidth= & 表格标题的宽度，默认为 4\,in \\
\end{longtable}
```

5-1-28

表 5.2　longtable 环境中的命令汇总

环境的水平对齐可选项	
留空	表格居中[①]
[c]	表格居中
[l]	表格左对齐
[r]	表格右对齐
结束表格一行的命令	
\\	普通的结束一行表格
\\[⟨距离⟩]	结束一行，并增加额外间距
*	结束一行，禁止在此分页
\kill	当前行不输出，只参与宽度计算
\endhead	此命令以上部分是每页的表头
\endfirsthead	此命令以上部分是表格第一页的表头
\endfoot	此命令以上部分是每页的表尾
\endlastfoot	此命令以上部分是表格最后一页的表尾
标题命令	
\caption{⟨标题⟩}	生成带编号的表格标题
\caption*{⟨标题⟩}	生成不带编号的表格标题
分页控制	
\newpage	强制分页
\pagebreak[⟨程度⟩]	允许分页的程度（0–4）
\nopagebreak[⟨程度⟩]	禁止分页的程度（0–4）
脚注控制	
\footnote	使用脚注[②]，注意不能用在表格头尾
\footnotemark	单独产生脚注编号，不能用在表格头尾

接下一页表格……

① 实际上，留空的对齐方式是由一组命令控制的，参见宏包文档。
② 普通表格中不能用。

（续表）

\footnotetext	单独产生脚注文字
长度参数	
\LTleft	对齐方式留空时，表格左边的间距，默认为 \fill
\LTright	对齐方式留空时，表格右边的间距，默认为 \fill
\LTpre	表格上方间距，默认为 \bigskipamount
\LTpost	表格下方间距，默认为 \bigskipamount
\LTcapwidth	表格标题的宽度，默认为 4 in

需要注意的是，longtable 得到的长表格，是把跨页时需要的长度信息保存在 .aux 辅助文件中，供下一次编译时控制表格宽度和表线。因此，一般需要编译两到三次，才能得到正确的表格。

ltxtable 宏包[43] 是 longtable 与 tabularx 宏包的结合体，使用它可以得到自动延伸的表列，同时也具有跨页功能。

ltxtable 宏包的用法比较特别，它需要在一个单独的 TEX 文件中编写 longtable 环境的表格，这个 longtable 环境可以使用 X 列格式符。然后，在实际的 TEX 源文件中使用命令 \LTXtable{⟨宽度⟩}{⟨文件名⟩} 来插入实际的表格。例如，我们文档的源文件是 foo.tex，那么里面可以使用这样的代码：

5-1-29
```
% foo.tex
% 导言区用：
\usepackage{ltxtable}
% 正文使用：
\LTXtable{\textwidth}{mytable}
```

然后在表格文件 mytable.tex 中，使用这样的代码：

5-1-30
```
% mytable.tex
\begin{longtable}{|X|X|X|}
  ...
\end{longtable}
```

由于编写的都是需要跨页的大型表格，因而将表格放在一个单独的文件中通常也是合适的。不过，有时我们也希望只在一个主文件中编写内容，而不需要将表格放在另一个文件中，ltxtable 宏包的这种用法还是太麻烦了。这时，可以借用 fancyvrb 宏包的 VerbatimOut 环境[301]，先生成单独的表格文件，然后再用 \input 命令读入排版表

格，把所有代码集中起来。在这种情况下，表格文件也可以在一次编译中重复使用，这对于需要排版大量类似表格的文档可能更为有用，例如：

```
% 导言区使用
% \usepackage{ltxtable}
% \usepackage{fancyvrb}
\begin{VerbatimOut}{\jobname.vrb}
\begin{longtable}{|c|X|X|X|X|X|}
\caption{各种序号} \\ \hline
\endfirsthead
\hline
\endhead
\hline
\endfoot
  数字 & 1 & 2 & 3 & 4 & 5 \\ \hline
  字母 & A & B & C & D & E \\ \hline
  天干 & 甲 & 乙 & 丙 & 丁 & 戊 \\
\end{longtable}
\end{VerbatimOut}
\LTXtable{0.5\textwidth}{\jobname.vrb}
```

5-1-31

表 5.3　各种序号

数字	1	2	3	4	5
字母	A	B	C	D	E
天干	甲	乙	丙	丁	戊

在例 5-1-31 中，我们首先使用 VerbatimOut 环境把一个 longtable 宏包的内容保存在 \jobname.vrb 文件中，然后使用 \LTXtable 命令将文件中的表格读入排版。在这里 \jobname 是一个 TEX 内部变量，内容是当前主要的 TEX 源文档的文件名（不含 .tex 扩展名），如对于 foo.tex 源文件来说，将内容是 lontable 环境的生成 foo.vrb 文件，然后立即读入排版。当然，这里的这个例子只有短短几行，看不出来表格换页的效果。

除了上面的方法，也可以使用 tabu 宏包[54] 的 longtabu 环境来排版长表格。tabu 宏包整合了许多表格宏包，提供了方便的用户界面。tabu 本身也提供了 X 列格式，而

longtabu 则会调用 longtable 的功能，因而上面的例子也可以简单地写成：

```
% \usepackage{tabu}
% \usepackage{longtable} % 仍然需要载入 longtable
\begin{longtabu}to 0.5\textwidth{|c|X|X|X|X|X|}
\hline
\endhead
\hline
\endfoot
\caption{各种序号} \\ \hline
\endfirsthead
  数字 & 1 & 2 & 3 & 4 & 5 \\ \hline
  字母 & A & B & C & D & E \\ \hline
  天干 & 甲 & 乙 & 丙 & 丁 & 戊 \\
\end{longtabu}
```

5-1-32

除了 longtable，也可以使用 xtab 宏包[295] 提供的 xtabular 环境来排版跨页的长表格。可以使用 \topcaption、\bottomcaption 或 \tablecaption 来输出表格顶部、底部或默认位置的标题；而命令 \tablefirsthead、\tablehead、\tablelasthead 与 \tabletail、\tablelasttail 命令在功能上也与 longtable 的 \endhead 等命令对应，不过这些命令都用在 xtabular 环境外面，例如：

```
% \usepackage{xtab}
\begin{center}
\tablecaption{各种序号}
\tablefirsthead{\hline}
\tabletail{\hline \multicolumn{6}{r}{\small 接下页}\\}
\tablelasttail{\hline}
\begin{xtabular}{|*{6}{c|}}
  数字 & 1 & 2 & 3 & 4 & 5 \\ \hline
  字母 & A & B & C & D & E \\ \hline
  天干 & 甲 & 乙 & 丙 & 丁 & 戊 \\ \hline
\end{xtabular}
\end{center}
```

5-1-33

表 5.4　各种序号

数字	1	2	3	4	5
字母	A	B	C	D	E
天干	甲	乙	丙	丁	戊

xtab 也同时提供一个 xtabular* 环境作为定宽长表格，其用法与 tabular* 环境类似，不过与 tabular* 一样不易使用。xtab 宏包的优点是它也可以在双栏环境中正常使用（不包括 multicol 宏包），不过同样需要多次编译。

longtable 和 ltxtable 主要解决的是表格过长的问题，但实际中有时还会遇到过宽的表格。LaTeX 并没有提供对宽表格的特别处理，也缺少类似 longtable 这样的宏包可以把宽表格自动断开。因此，对付过宽的表格，一般只能绕过问题，减小字号和列间距，或把页面临时设置为横向的（参见 2.4.2、5.2.3 节），或是手工把表格分开来。实际中应该根据需要调整使用的方法。

5.1.5　三线表与表线控制

下面我们来仔细研究表格线。

在科技文档中，数据表经常被表现为一种只有三种横线的形式，即三线表。为了美观，三线表顶部和底部的两条横线比较粗，而中间分隔表头与数据的线则较细（见表 5.5）。

表 5.5　某校学生身高体重样本

序号	性别	年龄	身高/cm	体重/kg
1	F	14	156	42
2	F	16	158	45
3	M	14	162	48
4	M	15	163	50

三线表需要能使用粗线不同的表格线，这可以用 booktabs 宏包[74] 得到。booktabs 提供以下几个表线命令：

*本节内容初次阅读可略过。

- \toprule 命令用来画表格顶部的粗线，下方有少量垂直间距，可以带一个可选的参数改变画线的粗细。
- \midrule 命令用来画表格中间的细分隔线，上下有少量垂直间距，可以带一个可选的参数改变线粗细。
- \bottomrule 命令用来画表格底部的粗线，上方有少量垂直间距，可以带一个可选的参数改变线粗细。
- \cmidrule 的作用与 \cline 类似，可以画出比 \midrule 更细的分隔线，上下有少量垂直间距，并且可以指定横线所在的列。\cmidrule 通常用于分隔表头中的主项和子项。这个命令也可以带有可选的线粗细参数。

例如，要得到表 5.5，就可以用下面的代码制表：

```
\begin{tabular}{ccccc}
  \toprule
  序号 & 性别 & 年龄 & 身高/cm & 体重/kg \\
  \midrule
  1 & F & 14 & 156 & 42 \\
  2 & F & 16 & 158 & 45 \\
  3 & M & 14 & 162 & 48 \\
  4 & M & 15 & 163 & 50 \\
  \bottomrule
\end{tabular}
```

5-1-34

booktabs 宏包也提供了调整线宽的表线前后间隔的长度变量，它们是：

- \heavyrulewidth 设置 \toprule 和 \bottomrule 的粗细，默认 0.08 em。
- \lightrulewidth 设置 \midrule 的粗细，默认 0.05 em。
- \cmidrulewidth 设置 \cmidrule 的粗细，默认 0.03 em。
- \aboverulesep 设置 \bottomrule、\midrule 和 \cmidrule 之前的间距。
- \belowrulesep 设置 \toprule、\midrule 和 \cmidrule 之后的间距。
- \abovetopsep 和 \belowbottomsep 分别设置表格顶底两条线前后的间距，默认为 0。

\cmidrule 命令在连续使用时，可以使用一组圆括号括起来的参数 l、r 或 l{⟨距离⟩}、r{⟨距离⟩} 表示间距的表格线可以在左右向内缩短一小段。而如果在同一位置画出双线的效果，则在多组 \cmidrule 之间需要加 \morecmidrules 分隔，它会产生合适的竖直间距，例如：

```
% 导言区 \usepackage{multirow,booktabs}
\begin{tabular}{*{6}{c}}
\bottomrule
\multirow{2}*{姓名} & \multicolumn{2}{c}{文科} &
  \multicolumn{2}{c}{理科} & \\
\cmidrule(lr){2-3}\cmidrule(lr){4-5}\cmidrule(lr){6-6}
  \morecmidrules\cmidrule(lr){6-6}
& 历史 & 文学 & 物理 & 化学 & 总评 \\
\midrule
张三 & A & A & B & A & A \\
\bottomrule
\end{tabular}
```

5-1-35

姓名	文科		理科		
	历史	文学	物理	化学	总评
张三	A	A	B	A	A

有关 booktabs 宏包的功能大致就是如此，更多的说明和示例可参见文档 Fear [74]。

booktabs 宏包是专门为绘制三线表而设计的宏包，它的横线命令很复杂，也在前后具有不同的垂直间距，这同时破坏表格中竖线的效果。如果只是希望控制横向表格线的粗细，而并不打算有额外的特性，makecell 宏包提供的 \Xhline{⟨线宽⟩} 与 \Xcline{⟨起⟩-⟨止⟩}{⟨线宽⟩} 是一组很好的选择，例如，我们用 \Xhline 和 \Xcline 模仿 booktabs 的行为：

```
% 导言区 \usepackage{makecell}
\begin{tabular}{c|cc}
\Xhline{2pt}
自变量 & \multicolumn{2}{c}{因变量} \\
\Xcline{2-3}{0.4pt}
半径 & 周长 & 面积 \\
\Xhline{1pt}
1.00 & 6.28 & 6.28 \\
2.00 & 12.57 & 12.57 \\
3.00 & 18.85 & 28.27 \\
```

```
\Xhline{2pt}
\end{tabular}
```

5-1-36

自变量	因变量	
半径	周长	面积
1.00	6.28	6.28
2.00	12.57	12.57
3.00	18.85	28.27

垂直方向的粗线可以使用 array 宏包的列格式说明符 !{⟨表线⟩} 所定义的自定义表格竖线得到（参见 5.1.6 节）。为了得到合适粗细的竖线，这里使用了原始 TeX 标尺盒子命令 \vrule（参见 [126、 141]），例如：

```
% \usepackage{makecell}
\newcolumntype{V}{!{\vrule width 2pt}}
\begin{tabular}{Vc|ccV}
\Xhline{2pt}
自变量 & \multicolumn{2}{cV}{因变量} \\
\Xcline{2-3}{0.4pt}
半径 & 周长 & 面积 \\
\Xhline{1pt}
1.00 & 6.28 & 6.28 \\
2.00 & 12.57 & 12.57 \\
3.00 & 18.85 & 28.27 \\
\Xhline{2pt}
\end{tabular}
```

5-1-37

自变量	因变量	
半径	周长	面积
1.00	6.28	6.28
2.00	12.57	12.57
3.00	18.85	28.27

一般地，如果要在文档中整体改变表格线的粗细，可以修改 \arrayrulewidth 长度变量。

表格中也可以使用双线。在 booktab 中使用 \cmidrule 的双线已如前述，而标准 LaTeX 的 tabular 环境中，只要使用两个竖线 || 作列格式说明，或是连续地使用 \hline\hline，就可以画出双线。相邻表格双线的距离由长度变量 \doublerulesep 控制。例如：

```
\begin{tabular}{|c||cc|}
\hline\hline
自变量 & \multicolumn{2}{c|}{因变量} \\
\cline{2-3}
半径 & 周长 & 面积 \\
\hline\hline
1.00 & 6.28 & 6.28 \\
2.00 & 12.57 & 12.57 \\
3.00 & 18.85 & 28.27 \\
\hline\hline
\end{tabular}
```

5-1-38

自变量	因变量	
半径	周长	面积
1.00	6.28	6.28
2.00	12.57	12.57
3.00	18.85	28.27

不过正如例 5-1-38 中所看到的那样，表格中的双线在拐弯、接口处通常会变得很难看。由于效果原因，通常并不鼓励在表格中使用双线，如果确实需要对双线的接口进行设置，可以使用 hhline 宏包[40] 提供的 \hhline 代替两个 \hline，对所有双线接口一一处理。hhline 宏包的详细功能可参见文档，另外注意旧版本的 colortbl 宏包一起使用时，需要设置双线间的填充色为背景白色。不过，由于使用复杂，实用中最好能避免这种双线发生交叉的情形，这里只举一个例子：

```
% \usepackage{hhline}
\begin{tabular}{|c||cc|}
\hhline{|=:t:==|}
半径 & 周长 & 面积 \\
\hhline{|=::==|}
1.00 & 6.28 & 6.28 \\
```

```
\hhline{|=:b:==|}
\end{tabular}
```

5-1-39

半径	周长	面积
1.00	6.28	6.28

在本节的最后，我们来看表格中虚线的使用。arydshln 宏包[176] 提供了一系列表格虚线的功能，可以用于 tabular、array 或 longtable 等表格环境中。

垂直的虚线使用列格式符：表示，水平的虚线使用 \hdashline 和 \cdashline 命令表示（对应于 \hline 和 \cline），这是 arydshln 宏包的基本用法，例如用虚线分隔的增广矩阵：

```
% \usepackage{arydshln}
\[
\left(
\begin{array}{@{}ccc:c@{}}
  a_{11} & a_{12} & a_{13} & b_1 \\
  a_{21} & a_{22} & a_{23} & b_2 \\
  a_{31} & a_{32} & a_{33} & b_3 \\
  \cdashline{1-3}
  0 & 0 & 0 & b_4 \\
\end{array}
\right)
\]
```

5-1-40

$$\left(\begin{array}{ccc:c} a_{11} & a_{12} & a_{13} & b_1 \\ a_{21} & a_{22} & a_{23} & b_2 \\ a_{31} & a_{32} & a_{33} & b_3 \\ \hdashline 0 & 0 & 0 & b_4 \end{array}\right)$$

此外，如果同时使用了 array 宏包，则还可以使用 \firsthline 与 \lasthline 的虚线对应命令 \firsthdashline 与 \lasthdashline。

arydshln 宏包中虚线的格式可以使用 \dashlinedash 和 \dashlinegap 两个长度变量控制。它们分别表示虚线的黑白长度，默认值都是 4 pt。可以修改它们得到特别的虚线，如改为密集的点线风格：

```
% \usepackage{arydshln}
\setlength\dashlinedash{1pt}
\setlength\dashlinegap{2pt}
\begin{tabular}{:cc:cc:}
\hdashline
上 & 上 & 上 & 上 \\
\cdashline{1-2}
下 & 下 & 下 & 下 \\
\hdashline
\end{tabular}
```

5-1-41

上	上	上	上
下	下	下	下

进一步地，命令 \hdashline 和 \cdashline 的后面也可以使用 [⟨线⟩/⟨空⟩] 的格式设置某一条虚线的黑白长度，竖线则可以使用带一个参数的列格式说明符 ;{⟨线⟩/⟨空⟩}，例如：

```
% \usepackage{arydshln}
\begin{tabular}{;{8pt/2pt}cc;{2pt/2pt}cc;{8pt/2pt}}
\hdashline[8pt/2pt]
上 & 上 & 上 & 上 \\
\cdashline{1-2}[2pt/2pt]
下 & 下 & 下 & 下 \\
\hdashline[8pt/2pt]
\end{tabular}
```

5-1-42

上	上	上	上
下	下	下	下

注意 arydshln 中 : 和 ; 产生的竖线不能用在 @ 和 ! 格式符之后。另外 arydshln 宏包也可能会与 hhline、makecell 等有关表格线的宏包冲突，使竖线不正常，此时可以利用 array 宏包的功能重定义普通竖线列格式 |：

```
% 解决 arydshln 与 hhline, makecell 的冲突
\usepackage{array}
\newcolumntype{|}{!{\vline}}
```

5-1-43

有关 arydshln 宏包的更多注意事项和精细调整的内容可参见文档 Nakashima [176]。

tabu 宏包为 tabu 和 longtabu 环境提供了自己的虚线命令，功能上可以完全代替 arydshln，并且更为强大。读者可参见 Chervet [54] 了解 tabu 中的表线控制功能。

5.1.6 array 宏包与列格式控制

前面几节中我们已经看到，有关表格的许多宏包都依赖 array 宏包的功能。array 宏包是标准 LaTeX 2ε 中 tabular 与 array 的一个增强的实现。这里对 array 宏包的扩展做一个简单的介绍。

array 宏包最基本的功能是对 tabular 和 array 环境原有列格式的扩展。在 5.1.1 节已经说明了 l、c、r、p{⟨宽度⟩}、@{⟨内容⟩} 几种列格式以及画竖线的 |、表示重复的 *{⟨次数⟩}{⟨格式⟩}，而 array 宏包又增加了以下几种：

- m{⟨宽⟩} 类似 p 格式，产生具有固定宽度的列，并可以自动换行。相当于由 \parbox{⟨宽⟩} 得到，垂直方向居中对齐。
- b{⟨宽⟩} 类似 p 和 m 格式，只是垂直方向与最后一行对齐。相当于由 \parbox[b]{⟨宽⟩} 得到。
- >{⟨内容⟩} 把 ⟨内容⟩ 插入后面一列的开头。
- <{⟨内容⟩} 把 ⟨内容⟩ 插入前面一列的末尾。
- !{⟨内容⟩} 把 ⟨内容⟩ 作为表格线处理。相当于使用了 @{⟨内容⟩} 格式，但左右两边会有额外的间距。

格式符 > 与 < 通常用来设置整列的格式。例如改变一列表格的字体，或是在某一列中使用数学模式：

```
% \usepackage{array} 或调用其他依赖 array 的宏包
\begin{tabular}{>{\bfseries}c|>{\itshape}c>{$}c<{$}}
\hline
姓名 & \textnormal{得分} & \multicolumn{1}{c}{额外加分} \\
\hline
张三 & 85 & +7 \\
```

*本节内容初次阅读可略过。

```
李四 & 82 & 0 \\
王五 & 70 & -2 \\
\hline
\end{tabular}
```

5-1-44

姓名	得分	额外加分
张三	*85*	+7
李四	*82*	0
王五	*70*	−2

用 p、m、b 列格式符得到表列中，由于使用 \parbox 产生盒子，段落缩进默认为零（参见 2.2.8 节）。此时也可以使用 > 来增加段前的缩进，或是指定盒子内容的对齐方式，注意此时需要在 \centering、\raggedright 等命令后面增加 \arraybackslash 命令：

```
% \usepackage{array}
\begin{tabular}{|>{$}r<{$}|>{\setlength\parindent{2em}}m{15em}|%
  >{\centering\arraybackslash}m{4em}|}
\hline
\pi & 希腊字母，多用于表示圆周率，也常用做变量。表示圆周率时多使用
  直立体。 & 常用 \\
\hline
\aleph & 希伯来字母的第一个，在数学中通常用于表示特殊集合的基数。
  & 不常用 \\
\hline
\end{tabular}
```

5-1-45

π	希腊字母，多用于表示圆周率，也常用做变量。表示圆周率时多使用直立体。	常用
$\aleph$	希伯来字母的第一个，在数学中通常用于表示特殊集合的基数。	不常用

格式符！则用来产生特殊表格线，5.1.2 节中利用原始标尺盒子命令 \vrule 得到粗表格线就是一个例子。这里再给出一个使用特殊符号代替表格线的例子：

5-1-46

```
% \usepackage{array}
\begin{tabular}{c!{$\Rightarrow$}c}
张三 & 85 \\
李四 & 82 \\
王五 & 70 \\
\end{tabular}
```

张三 ⇒ 85
李四 ⇒ 82
王五 ⇒ 70

array 宏包大大扩展了表格列格式的使用，但同时也让表格的列格式变得复杂而难以辨认，对于例 5-1-45 这类复杂表格则更为严重。为此，array 宏包还提供了 \newcolumntype 命令，用来定义新的列格式说明符，简化复杂的列格式。\newcolumntype 命令的语法与 \newcommand 命令非常类似，如例 5-1-45 中的几个列格式就可以这样定义：

5-1-47

```
% \usepackage{array}
\newcolumntype{M}{>{$}c<{$}}
\newcolumntype{P}[1]{>{\setlength\parindent{2em}}p{#1}}
\newcolumntype{C}[1]{>{\centering\arraybackslash}m{#1}}
% 使用新的列格式：
\begin{tabular}{|M|P{15em}|C{4em}|}
  ...
\end{tabular}
```

也可以用 \newcolumntype 命令定义多字符列格式，不过通常都只使用一个字符。

array 宏包提供了对表格整列进行格式设置的方法。但标准的 LaTeX 表格不能对一行的整体格式进行统一设置。为此，可以使用 tabu 宏包的 \rowfont 命令设置 tabu 环境中一整行的字体，例如：

```
% \usepackage{tabu}
\begin{tabu}{ccc}
\hline
\rowfont{\bfseries}
姓名 & 得分 & 额外加分 \\
\hline
张三 & 85 & $+7$ \\
\rowfont{\itshape}
李四 & 82 & 0 \\
```

```
王五 & 70 & $-2$ \\
\hline
\end{tabu}
```

5-1-48

姓名	得分	额外加分
张三	85	+7
李四	*82*	*0*
王五	70	−2

练 习

5.7 尝试仅使用 array 宏包，实现 tabu 宏包中 \rowfont 对表格整行设置字体的效果。

5.1.7 定界符与子矩阵

在 4.2.5 节中我们已经看到，数学矩阵可以由 amsmath 宏包提供的 matrix 环境得到，而两边带有定界符的矩阵则可以由 pmatrix、bmatrix 等一系列环境得到。更一般的定界符则可以通过在 matrix 环境两边使用 \left、\right 命令加上原始定界符获得，例如：

```
\[ \left\{ \begin{matrix}
  1 & 2 \\ 3 & 4
\end{matrix} \right. \]
```

$$\begin{cases} 1 & 2 \\ 3 & 4 \end{cases}$$

5-1-49

事实上，amsmath 中的 matrix 环境就实现为一个列格式固定居中的 array 环境[167]，其等价定义就是：

```
\newcounter{MaxMatrixCols}
\setcounter{MaxMatrixCols}{10}
\newenvironment{matrix}
  {\begin{array}{@{} *{\value{MaxMatrixCols}}{c} @{}}}
  {\end{array}}
```

*本节内容初次阅读可略过。

因而各种矩阵都可以使用 array 环境实现。

delarray 宏包[39] 扩展了标准的列格式说明语法，可以在列格式说明列表两侧直接给出定界符，而不必再在 array 环境两边加 \left, \right。这在需要大量书写特殊定界符的矩阵时非常方便，例如：

5-1-50

```
% \usepackage{delarray}
\[
  \begin{array}{cc}]   % 左边圆括号，右边方括号
    1 & 2 \\
    3 & 4
  \end{array}
\]
```

$$\begin{pmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$$

delarray 宏包会与 arydshln, colortbl 等其他一些影响表格格式的宏包冲突，而并不增加新的功能，所以往往并不直接使用。但它的这种语法却影响了其他表格工具宏包，例如，tabu 宏包在使用 delarray 选项后就支持相同的语法：

5-1-51

```
% \usepackage[delarray]{tabu}
\[  \begin{tabu}{cc}
    1 & 2 \\
    3 & 4
\end{tabu}  \]
```

$$\begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix}$$

另一个工具是 blkarray 宏包[36]，它提供了 blockarray 环境，可以代替文本模式的 tabular 环境及数学模式的 array 环境。命令 \BAmulticolumn 对应于标准 LaTeX 的 \multicolumn，命令 \BAnewcolumntype 则对应于 array 宏包的 \newcolumntype。blockarray 环境扩展了列格式说明符，可以在列格式设置中加上 & 符号，表示将其后的 @ 说明或表格线说明 | 看成是下一列左边的内容，以方便 \BAmulticolumn 临时修改，例如：

```
% \usepackage{blkarray}
% 如果不用 &| 说明，则竖线 | 将会被看成是中间一列的内容
\begin{blockarray}{|l|c&|r|}
```

```
  张三 & Zhang & 80 \\
       % 不用在 r 后面用 |，也不影响表格线
  李四 & \BAmulticolumn{1}{r}{Li} & 78 \\
  王五 & Wang & 100 \\
\end{blockarray}
```

5-1-52

张三	Zhang	80
李四	Li	78
王五	Wang	100

blockarray 环境在数学模式中使用时，就可以将定界符的说明直接放在列说明符内，例如：

```
% \usepackage{blkarray}
\[  \begin{blockarray}{(cc]}
  1 & 2 \\
  3 & 4
\end{blockarray}  \]
```

$$\begin{pmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$$

5-1-53

blkarray 宏包更重要的功能是其 block 环境，它用在 blockarray 环境内部，用于把矩阵中的几行分开，成为单独的小块。可以单独设置这一块的对齐方式、表格线、定界符等。可以把 block 环境看做是 \BAmulticolumn 命令的推广形式，对应的星号环境 block* 则禁用本身的定界符设置，让外层 blockarray 环境设置的定界符起效。这特别适合复杂的带定界符的分块矩阵的书写，例如：

```
% \usepackage{blkarray}
\[  \left[
\begin{blockarray}{*4r}
\begin{block}{(rr)rr}
 a & -b & 0 & 0 \\
-c &  d & 0 & 0 \\
\end{block}
\begin{block}{rr(rr)}
0 & 0 & -a &  b \\
0 & 0 &  c & -d \\
```

```
\end{block}
\end{blockarray}
\right]  \]
```

5-1-54

$$\begin{bmatrix} \begin{pmatrix} a & -b \\ -c & d \end{pmatrix} & \begin{matrix} 0 & 0 \\ 0 & 0 \end{matrix} \\ \begin{matrix} 0 & 0 \\ 0 & 0 \end{matrix} & \begin{pmatrix} -a & b \\ c & -d \end{pmatrix} \end{bmatrix}$$

\BAmultirow 命令则可以用在 block 的列格式说明符中，或是表格内部，表示跨行的子表格内容，这对于花括号特别有用，例如：

```
\[
\begin{blockarray}{ccc}
\begin{block}{cc\}\BAmultirow{4em}}
1 & 2 & 自然数 \\
3 & 4 & {} \\ % 空白 {} 占位
\end{block}
\begin{block}{cc\}1}
-1.5 & \frac12 & \BAmultirow{4em}{实数} \\
3.5 & 40 & \\
\end{block}
\end{blockarray}
\]
```

5-1-55

$$\begin{matrix} 1 & 2 \\ 3 & 4 \\ -1.5 & \frac{1}{2} \\ 3.5 & 40 \end{matrix} \begin{matrix} \left.\begin{matrix} \\ \\ \end{matrix}\right\} \text{自然数} \\ \left.\begin{matrix} \\ \\ \end{matrix}\right\} \text{实数} \end{matrix}$$

非常遗憾的是，blkarray 宏包产生的较早，却与重要的 amsmath 宏包有冲突，在一起使用时可能出现编译错误。该宏包目前基本停止维护，不过可以手工在导言区使用如下方式临时解决此冲突：

```
\usepackage{blkarray}
\makeatletter
\newbox\BA@first@box
\makeatother
```

5-1-56

有关 blkarray 宏包的其他功能和进一步使用事项，可参见宏包手册 Carlisle [36] 的说明和示例。

练 习

5.8 在 4.2.5 节中，我们看到了原始命令 \bordermatrix 可以用来输入带有边注的矩阵。试说明如何使用 blkarray 宏包的功能代替和扩展 \bordermatrix，并利用 blkarray 宏包，排版下面带说明的矩阵：

$$\begin{array}{c} \\ 1 \\ 2 \\ \\ \end{array}\!\!\begin{array}{c} \begin{array}{cc} 1 & 2 \end{array} \\ \begin{bmatrix} \alpha & \beta \\ \gamma & \delta \end{bmatrix} \\ \uparrow \\ j \end{array}\!\!\begin{array}{l} \\ \leftarrow i \\ \\ \\ \end{array}$$

5.2 插图与变换

按图形的生成机制区分，在 LaTeX 中使用的图形大约有以下三类[84]：

1. 使用 TeX 的基本命令。不过原始的 TeX 本身提供的绘图功能非常少，只能做到精确定位和画水平、垂直方向的粗细线条（标尺盒子），完全没有画斜线、曲线、填充、彩色之类的功能。例如 LaTeX 标准命令中就以这种机制使用描点的方式画曲线。不过这种方式的能达成的效果非常少，质量也不好，现在也很少使用了。
2. 使用特殊的字体拼接组合，即事先做好一些作为图形零件的字体符号，使用 TeX 将它们准确地拼接在一起。事实上许多数学公式中的符号（如 \overbrace）就是用这种方法得到的，LaTeX 的标准命令用这种方式画倾斜的直线、圆弧、箭头；XY-pic 宏包（参见 5.5.1 节）则使用这种方式画更复杂的数学图形。不过，这种方式的绘图能力也相当有限，能得到的图形依赖于字体符号表，也缺少填充、彩色之类的功能，现在也已经更多地转向其他方式了。

3. 最后一种方式是脱离原本 Knuth TEX 的功能，利用 TEX 的扩充接口 \special 命令或新引擎的功能，直接输出 PostScript 或 PDF 的图形。这种图形也有两个子类别：一种是使用 PostScript 等语言在输出文件中画图，另一种则是向 PostScript、PDF 文件中插入其他类型的图形。这类方式失去了 DVI 格式输出原有的“设备无关性”，输出图形的对象也从 DVI 文件转向了 PostScript、PDF 等格式。另外，也由此可以借助 PostScript、PDF 等输出格式的能力，突破 DVI 文件的限制，得到高精度的曲线、彩色、透明等等图形能力。使用输出引擎特定的功能画图或插图，这也是现代 LATEX 文件最主要的图形使用方式。

在这一节里，我们将看到在 LATEX 中插入外部图形或做几何变换的方式。尽管使用的 TEX 引擎或输出驱动有所不同，但 LATEX 的基本宏包 graphics 或 graphicx 为插图和几何变换提供了统一的命令，使我们可以像普通字符一样对待图形。

5.2.1 graphicx 与插图

插图功能是利用 TEX 的特定编译程序提供的机制实现的，不同的编译程序支持不同的图形方式。这个事实常常令人不知所措，因为在不同年代和背景的书籍、文档中往往会有大相径庭的说法，其中最流行的（也是不准确的）说法是“LATEX 只支持 EPS”图形，好在常用编译程序所支持的图形格式是比较接近的，见表 5.6。其中 METAPOST 输出的 MPS 格式可以看做特殊的一类 EPS，因此支持 EPS 的编译方式实际也都支持 MPS 格式的图形，参见 5.5.3.1 节。

在 LATEX 中，插图是由 graphics 或 graphicx 宏包[38] 所使用的 \includegraphics 命令完成的。例如我们已经有一幅名为 lion.eps 的狮子图形，那么插图只要一条简单的命令：

5-2-1

```
% 导言区 \usepackage{graphics}
% 或 \usepackage{graphicx}
狮子：\includegraphics{lion}
```

狮子： 

graphics 宏包与 graphicx 宏包在功能上并没有什么差别。graphicx 宏包支持 ⟨项目⟩=⟨值⟩ 的语法，使用起来更为方便，因此本节下面的介绍都以 graphicx 宏包为准，以下都假定文档中已经使用了

```
\usepackage{graphicx}
```

表 5.6　LaTeX 编译程序与插图格式

TeX 引擎命令	图形驱动	支持的格式	备注
latex	Dvips	EPS	MiKTeX 还部分支持 PNG 和 JEPG
latex	DVIPDFMx	EPS, PDF, PNG, JPEG	PDF、PNG、JEPG 需要使用 extractbb 程序生成 .xbb 文件
pdflatex		MPS, PDF, PNG, JPEG	MPS 是 METAPOST 的输出格式，TeX Live 2010 以后还能自动将 EPS 转换为 PDF 文件插入
xelatex	xdvipdfmx	EPS, PDF, PNG, JPEG, BMP	驱动是自动调用的，MAC 系统下的旧驱动 xdv2pdf 还支持其他一些格式

插图的核心命令是 \includegraphics，其语法格式如下：

\includegraphics[⟨选项⟩]{⟨文件名⟩}

其中 ⟨文件名⟩ 是图形文件的文件名，一般文件的扩展名可以省略不写，LaTeX 会自动查找它支持的文件格式，为了明确也可以加上扩展名。

插入的图形都有一个自然比例，对于 EPS、PDF 图形就是制作的尺寸，对于 JPG、PNG、BMP 等像素图的尺寸则是点阵数除以图形打印度（一般用每英寸点数 DPI 表示）。可以给 \includegraphics 命令加一些可选项来调整图形的大小、位置等，如可以用命令选项 width、height 和 scale 设置图形的宽度、高度或缩放比例，这也是最常用的几个插图选项，例如：

```
\includegraphics[width=2em]{lion}
\includegraphics[height=1cm]{lion}
\includegraphics[scale=0.5]{lion}
```

5-2-2

使用 angle 选项可以让图形逆时针旋转一定角度，旋转的中心可以用 origin 选项确定。origin 的值可以用字符 l, r, c, t, b, B 中的一个或两个，分别表示左、右、中、

上、下和基线，默认值是 lb。例如：

5-2-3
```
旋转的狮子：
\includegraphics[angle=90]
  {lion.eps}
\includegraphics[angle=-45,origin=c]
  {lion.eps}
```

旋转的狮子： 

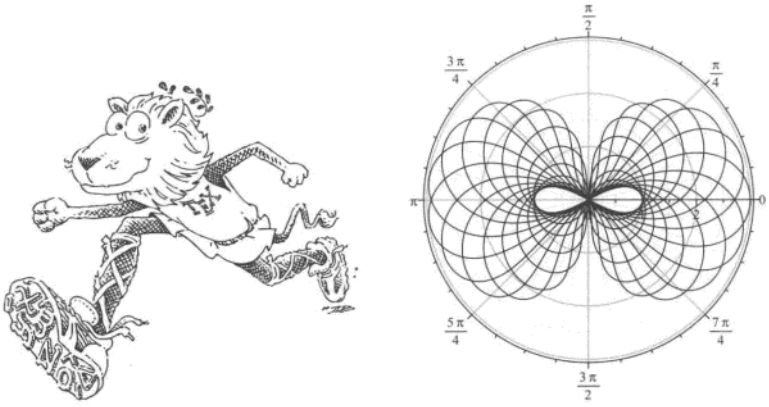
插入图片的基线就在图片的底部，因而图片盒子的深度为零。但图片旋转时，基线和深度会随 origin 不同而变化，此时要对图片放缩，就要用 totalheight 代替 height 选项，表示规定图片深度与高度之和，例如：

5-2-4
```
基线\rule{2cm}{0.4pt}%
\includegraphics[angle=90,origin=b,
  totalheight=1.5cm]{lion.eps}
```

基线 

如前所述，除非使用传统的 Dvips 程序作为图形驱动，一般编译程序支持的图形格式都有好几种，实际插入图形格式也就有不同的选择。对于 pdfLaTeX、XeLaTeX 这些现代的常用编译程序来说，PDF、PNG、JPEG（扩展名是 .jpg 或 .jpeg）这三种格式是更为常用的，它们的用途也各自不同：

- PDF 图片通常用来作为矢量图形的标准格式。矢量图形的可以任意比例放缩而不影响输出效果，在表现固定图案或数据产生的图形时很有优势。在以前 LaTeX 更多使用 EPS 格式的矢量图形，不过现在支持输出 PDF 的作图软件变得更加普及，而且 PDF 格式的文件通常比相同内容的 EPS 图形体积小，功能（如透明色）也可能更多。一般的矢量图设计软件（如 Illustrator、CorelDraw、Inkscape）、专业数学软件（如 MATLAB、Maple、Mathematica），计算或作图语言（如 R、GNUplot），图论或流程图工具（如 Visio、Dia、Graphviz），物理、化学或工程图工具（如 JaxoDraw、ChemDraw、AutoCAD）等等，都可以保存或打印为 PDF 格式的图片供 LaTeX 使用（见图 5.1）。



(a) 矢量格式的 TEX Live 吉祥物（作者 Duane Bibby）　　(b) 使用 Maple 画的极坐标函数图形

图 5.1　适合使用 PDF 格式表现的图形

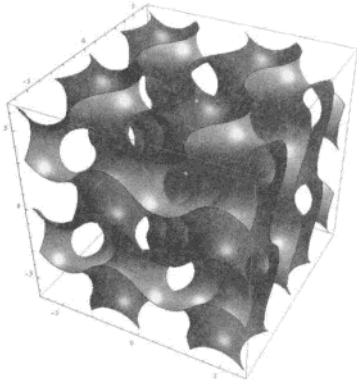
- PNG 图片是无损压缩的像素图格式。通常用来显示计算机制作的非自然图形，如复杂数据可视化的结果。一般能输出矢量图的软件也都可以输出 PNG 格式的像素图，也有一些科技作图软件不支持矢量图的。通常数据可视化的图形最好使用矢量格式，不过对于一些特殊情形，如逐点产生的动力系统图像、3D CG 图，还是适合使用 PNG 格式（见图 5.2）。
- JEPG 图片是有损压缩的像素图格式，通常用作照片的格式（见图 5.3）。计算机产生的非自然图形最好不要使用这种有损压缩格式。

有些时候，我们不能完全自己决定使用哪种编译程序编译文档，例如一些陈旧投稿系统可能只支持 Dvips 作为输出驱动。也有的时候，手边的作图软件也不能输出需要的图形格式，例如某些旧的 Windows 程序只支持 BMP 格式的图像输出。在这些情况下，我们就不得不对图形的格式进行适当的转换，以符合要求。

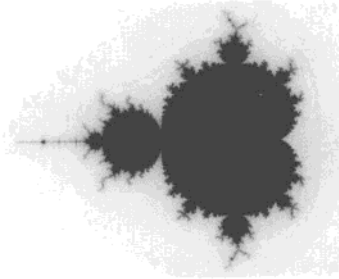
有大量的软件提供图形格式转换功能。绝大多数图像处理软件，从简单的 Windows 画图程序到专业的 Adobe Photoshop，都支持丰富的像素图像格式，很容易另存为 PNG 或 JPG 格式。而如果有大量图像需要批量转换，则可以使用 ImageMagick 这类命令行工具（参见 1.1.2.3 节）。例如，下面的 Windows 控制台命令可以把当前目录下的所有 BMP 文件转换为 PNG 文件：

```
for %I in (*.bmp) do convert %I %~nI.png
```

从 EPS 或 PDF 格式向 PNG 或 JPEG 的转换，也可以调用 GhostScript 来完成（参见 1.1.2.3 节）。例如在 Windows 下把所有 PDF 文件以 96 DPI 的精度转换为 PNG 文件：



(a) 使用 Mathematica 绘制的特殊方程边界



(b) 使用 C 语言编程绘制的 Mandelbrot 集

图 5.2 适合使用 PNG 格式表现的图形

图 5.3 适合使用 JPEG 格式表现的图形——照片

```
for %I in (*.pdf) do (
 gswin32c -sDEVICE=png256 -dEPSCrop -r 96 -dTextAlphaBits=4 -o %~nI.png %I
)
```

有关 Windows 命令行的 for 语句，可参见 Windows 的联机帮助文件。事实上，还可以把这些内容预先编写成为 Windows 批处理，然后点击图标调用。

如果需要把 JPEG 或 PNG 等像素图转换为 EPS 格式，可以使用 TEX 系统附带的命令行工具 bmeps。bmeps 默认会将颜色图像转换成黑白格式的，如果要保留彩色，可以加选项 -c。例如将 JPEG 图像 foo.jpg 转换为 EPS 格式，可以用：

```
bmeps foo.jpg foo.eps
```

TEX 发行版附带的 bmeps 工具版本较老，得到的图像体积太大，JPEG 图像转换的效果也不好。TEX Live 的用户可以使用自带的 sam2p 工具完成转换，如：

```
sam2p foo.jpg foo.eps
```

如果要使用 ImageMagick 完成 JPEG/PNG 到 EPS 的转换工作，需要指定 eps3 格式，以避免产生体积过大的图形，如：

```
convert foo.jpg eps3:foo.eps
```

将 PDF 格式的图片转换为 EPS 格式也可以使用 GhostScript 完成。在 Windows 下转换 foo.pdf 到 foo.eps 可以用：

```
gswin32c -sDEVICE=epswrite -dEPSCrop -o foo.eps foo.pdf
```

当然这个转换也可以使用 ImageMagick 完成，不过 ImageMagick 主要是像素图的转换程序，涉及 EPS 和 PDF 格式的转换工作，用 GhostScript 和 sam2p 等专门工具处理比用 ImageMagick 问题更少一些。

转换图形格式的基本原则是避免转换，图形在多次转换过程中可能会损失精度，如 JPEG 的有损压缩，特别是要避免把矢量图转换为像素图，实践中一般只需要很有限的几类转换：

- 使用 Dvips 驱动时，需要把 PDF 矢量图、PNG 或 JPEG 像素图使用 GhostScript、sam2p 等工具转换为 EPS 格式。这对于一些陈旧的西文投稿系统仍然是必要的。
- 使用 pdfLaTEX 引擎时，需要把 EPS 矢量图用 GhostScript、ImageMagick 等工具转换为 PDF 格式。在 TEX Live 2010 默认会自动调用 epstopdf 宏包[184] 进行这种转换。
- 使用图像处理软件或 ImageMagick 把其他像素图格式（如 BMP、TIF）转换为 PNG 或 JPEG 格式。

从表 5.6 中可以看出，在使用 DVIPDFMx 作为输出驱动时，需要为非 EPS 图形使用 extractbb 生成 .xbb 文件。事实上，此时处理插图时，必须首先在调用 graphicx 宏包时使用 dvipdfmx 选项：

```
\usepackage[dvipdfmx]{graphicx}
```

然后命令行下使用命令[①]：

```
extractbb -x ⟨图形文件名⟩
```

生成与图形文件同名的 .xbb 文件，最后在文档中使用 \includegraphics 插入图形。.xbb 文件是一个文本文件，里面保存着图形的边界框坐标信息。例如我们使用 extracolsep -x photo.jpg 来处理图 5.3 中的照片，我们将得到这样格式的文件 photo.xbb：

```
%%Title: ./photo.jpg
%%Creator: extractbb 20100328
%%BoundingBox: 0 0 288 216
%%HiResBoundingBox: 0.000000 0.000000 288.000000 216.000000
%%CreationDate: ...
```

把它与照片文件放在一起，就可以正确编译了。在 TeX Live 2010 中，调用 extractbb 程序可以在编译时一次完成，此时需要在编译时给 latex 命令增加 -shell-escape 选项。如果觉得调用 extractbb 过于麻烦，也可以使用 bmpsize 宏包[182] 获取像素图的大小，如：

```
% 导言区
\usepackage[dvipdfmx]{graphicx}
\usepackage{bmpsize}
% 正文插图
\includegraphics{photo.jpg}
```

MiKTeX 的 Dvips 引擎支持 PNG 和 JPEG 图像（但会失去彩色），也需要使用适当的选项配合 .xbb 文件决定图像大小，此时使用 bmpsize 宏包也是比较方便的方式。

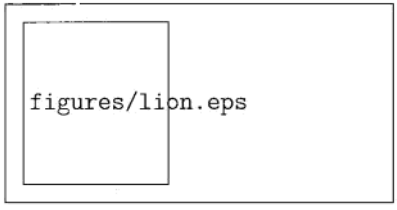
*本节剩余内容初次阅读可以略过

① 有的系统有用 ebb 或 xbb 命令表示 extractbb 程序的不同选项。

正如前面看到的，graphicx 宏包为特定的图形驱动提供了一些宏包选项，来决定内部使用什么方式处理图形。可用的驱动选项包括 dvips、dvipdfmx、pdftex、xetex 等。不过，Dvips 不使用特别的选项一般也能正确处理，pdfTEX 和 XETEX 引擎可以由系统自动识别，所以只有使用 latex + dvipdfmx 命令编译时，才必须使用 dvipdfmx 选项。DVIPDFMx 驱动原本是为了方便处理中日韩文字而设计的，随着 pdfTEX 和 XETEX 的发展，现在它对中文处理的重要性已经下降。DVIPDFMx 的插图显得有些麻烦，因而现在直接使用 XETEX 或 pdfLATEX 更为方便。

graphicx 宏包可以使用 draft 和 final 选项，用来表示是否实际插图。当宏包使用 draft 选项，或是文档类使用的全局的 draft 选项时，\includegraphics 命令并不会实际插入图形，而只是得到一个与图形大小相同的带有文件名的方框：

```
% 导言区 \usepackage[draft]{graphicx}
\includegraphics{lion}
```



5-2-5

这个选项对于未完成的草稿可以一定程度上加快编译，并减小文件大小。

图形文件一般与 .tex 源文件在同一目录中编译，也可以在 \includegraphics 命令中使用图片的相对路径或绝对路径插入图片。Widows 和 UNIX/Linux 系统的路径一律使用斜线分隔，例如：

```
\includegraphics{eps/foo.eps}       % 相对路径
\includegraphics{D:/photos/bar.jpg} % 绝对路径
```

graphicx 宏包一个特别有用的命令是 \graphicspath，它可以指定图形文件的搜索目录列表，不同的目录用分组隔开，例如：

```
\graphicspath{{figures/}} % 本书的设置，图片在当前目录下的 figures 目录
\graphicspath{{pdf/}{png/}{jpg/}} % 按图片类型管理的
```

5-2-6

除了前面介绍的控制大小和旋转的选项，\includegraphics 还有许多其他的选项，下面是完整的选项列表：

☞ bb

手工指定图形的边界框（bounding box）左下角和右上角的坐标，这对坐标会在 EPS 文件开头和 .xbb 文件中输出，通常不用手工输入。坐标用空格分隔的 4

个长度值表示，如果省略单位，则使用 bp。这个选项可以用来代替 extractbb 生成的 .xbb 文件的功能。

☞ **bbllx,bblly,bburx,bbury**

分别指定 bb 选项的四个分量。

☞ **natwidth,natheight**

使用 natwidth=⟨宽⟩, natheight=⟨高⟩ 就相当于设置了 bb=0 0 ⟨宽⟩ ⟨高⟩。

☞ **hiresbb**

一个真假值，true 或 false，true 可省略。表示用 EPS 文件中的元注释 %%HiResBoundingBox 代替 %%BoundingBox 表示边界框的大小，对一些软件生成的 EPS 文件有用。

☞ **viewport**

与 bb 类似，只是它是相对于原来定义边界框左下角的坐标，当原来的边界框左下角坐标是 (0,0) 时就和 bb 相同。如果不是要使用整个图片，而只是使用图片的一部分，就可以使用 viewport 选项截取原图的一部分。

☞ **trim**

四个长度值，用来表示在原来的边界框基础上向内（左、下、右、上）去掉的长度，与 viewport 选项一样可以起到截取部分图片的作用。

☞ **angle**

旋转的角度。

☞ **origin**

旋转的中心。

☞ **clip**

一个真假值，true 或 false，true 可省略。使用前面的选项定义边界框后，设置 clip 选项为真，就可以只截取图片的一部分；而默认值为假，则表示图片仍完整显示，只是图片盒子占用的位置按边界框计算。并非所有编译程序都支持 clip 选项，目前 XeTeX 就暂不支持。

☞ **width,height,totalheight**

把图片放缩到指定的宽度、高度或总高度。

☞ **keepaspectratio**

一个真假值，决定放缩时是否保持图片原有的长宽比。如果只设置了 width、height 或 totalheight 中的一个，默认为真。

☞ **scale**

设置放缩的比例，不放缩值为 1。

☞ **draft,final**

与宏包的选项相同。

☞ **type,ext,read,command**

设置图片类型、扩展名、边界框文件扩展名（如前面的 .xbb）、需要执行的特殊命令。这几个选项一般不会用到，详细说明参见 Carlisle [38, § 4.5]，这里不再做进一步地说明。

\DeclareGraphicsRule 命令用来声明对特定扩展名的图形文件的处理方式。分别对应于前面 ext, type, read, command 选项，其语法格式如下：

\DeclareGraphicsRule{⟨扩展名⟩}{⟨类型⟩}{⟨边界框扩展名⟩}{⟨特殊命令⟩}

在 5.5.3.1 节中可以看到相关的例子。

\DeclareGraphicsExtensions 命令用来声明在 \includegraphics 命令中可以省略不写的文件扩展名。其参数是一个扩展名列表。实际上相关列表在 graphics 宏包在载入图形驱动时就已设置完毕，一般用户不再使用。

有关图片插入的配置命令以及 graphics 宏包的语法，可进一步参见 Carlisle [38]。

5.2.2　几何变换

graphicx/graphics 宏包不仅提供了插图的功能，也提供了对任意对象（也就是盒子）进行几何变换的功能。除了平移，TEX 本身并没有几何变换的功能，这些功能同样是依赖具体的编译程序的。

\scalebox 命令用来按比例对内容放缩，其语法格式如下：

\scalebox{⟨水平因子⟩}[⟨垂直因子⟩]{⟨内容⟩}

其中 ⟨垂直因子⟩ 省略时，与 ⟨水平因子⟩ 相同，例如：

```
\scalebox{2}{大字}
\scalebox{2}[1]{扁字}
\scalebox{1}[2]{长字}
```

大字 扁字 长字

5-2-7

*本节内容初次阅读可略过。

\reflectbox{⟨内容⟩} 相当于 \scalebox{-1}[1]{⟨内容⟩}，用来对内容做水平镜像反射，例如：

5-2-8

```
\LARGE 汉字\reflectbox{汉字}
```

汉字字汉

\resizebox 命令用来把内容放缩到指定的宽度和高度，其语法格式如下：

\resizebox{⟨宽度⟩}{⟨高度⟩}{⟨内容⟩}

这里 ⟨宽度⟩ 和 ⟨高度⟩ 可以使用一个叹号 ！表示按比例随另一个分量放缩，带星号的 \resizebox* 命令功能相同，只是第二个参数表示盒子高度与深度之和，例如：

5-2-9

```
\resizebox{2cm}{1cm}{扁}
\resizebox{!}{1cm}{\fbox{高 1cm}}
```

扁
高 1cm

\rotatebox 命令用来对内容进行旋转，其语法格式如下：

\rotatebox[⟨选项⟩]{⟨角度⟩}{⟨内容⟩}

⟨选项⟩ 是 graphicx 宏包才有的。在 ⟨选项⟩ 中，可以使用 origin 选项表示旋转不动点，值与 \includegraphics 中的选项相同，使用字母 lrctbB 中的一个或两个，也可以使用 x=2mm,y=5mm 这样的选项准确指定旋转的不动点。units 选项是一个数字，表示旋转的单位，默认值为 1，也就是逆时针的 1°，设置 units=6.283185 就相当于使用弧度单位。下面举一个例子：

5-2-10

```
\rotatebox[origin=c]{90}{旋}%
\rotatebox[origin=c]{90}{转}%
的汉字
```

旋转的汉字

在旋转符号时，大多使用中心旋转。

graphicx 宏包并不提供错切变换的功能。如果有这种需要，可以使用 PSTricks、tikz 等绘图宏包（参见 5.5.2 节）。

5.2.3 页面旋转

在 LaTeX 中，不仅可以对盒子旋转，也可以对整个页面的内容进行旋转。lscape 宏包[37] 的 landscape 环境就可以把整个页面旋转 90°，如本节的内容就是这样得到的：

```
% 导言区 \usepackage{lscape}
\begin{landscape}
\subsection{页面旋转}
.....
\end{landscape}
```

5-2-11

在生成 PDF 文件的情况下，可以使用 pdflscape 宏包[181] 代替 lscape 宏包，它可以在旋转页面的同时，让输出的 PDF 文件的页面在阅读器中显示时同时顺时针旋转 90°，这样看到的文字就仍然是正确方向的，只是显得页面横过来了而已。

landscape 环境旋转的页面不会影响版心外面页眉页脚的输出，它通常用来表示过宽的内容，如大幅的图片或宽大的表格。不过，landscape 环境会在前后造成额外的分页，这对于只想插入一个图片或表格的情形，往往并不合适。理想的办法是产生旋转的浮动环境，为此，可以使用 rotfloat 宏包[237] 提供的 sidewaystable 和 sidewaysfigure 环境来插入浮动的表格或图形，同时不会影响浮动页前后的内容。rotfloat 宏包基于 rotating[72] 宏包和 float[149] 宏包，它会将所有浮动环境增加一个 sideways 开头的旋转环境，沿用图 5.3 的图片：

```
\begin{sidewaysfigure}[p]
  \centering
  \includegraphics[width=7in]{photo.jpg}
  \caption{贵州少数民族地区剪影}
\end{sidewaysfigure}
```

5-2-12

图 5.4 贵州少数民族地区剪影

5.3　浮动体与标题控制

图片或表格通常都占有较大的一块，直接放在文档中常常会造成分页的困难，即在前一页放不下，放在后一页又会造成很大的留白。其他一些大块的内容也可能有类似的问题，如程序算法、大型公式和不宜断开的特殊形状段落。LaTeX 对这个问题的解决方案是使用“浮动体”（float）。浮动体是一个活动的盒子，它可以把内容放在距离浮动体代码前后不远的地方，通常就是浮动体代码所在地，但也可以放在页面开头、末尾或是单独的一页中。使用浮动体，就可以在不太费力仔细调整内容的情况下，避免大块图表把整齐的页面弄糟。当然有时仍然需要手工的精细控制，才能得到良好的图文混排效果。

浮动体的另一个重要用途是给图表添加一个标题。LaTeX 的浮动体环境为图表标题提供了专门的命令进行自动编号、自动生成目录的功能，通过第三方宏包还可以对标题的格式进行整体设计。

这一节我们就将认识 LaTeX 的浮动体机制，解析浮动体及其标题设置。

5.3.1　浮动体

LaTeX 2_ε 的标准文档类预定义了两种浮动体环境：figure 和 table，通常分别用于图和表的排版。figure 环境的语法格式如下：

```
\begin{figure}[⟨允许位置⟩]
    ⟨任意内容⟩
\end{figure}
```

table 环境与之类似。其中可选参数⟨允许位置⟩用来设定浮动环境可以出现在页面的位置，即 h、t、b、p 四个选项的组合：

- **h**　此处（here），浮动体的内容被放在代码所在的上下文位置。
- **t**　页顶（top），浮动体被放在一页的顶部，这可以是代码所在环境的页面或之后的页面，注意当页排版的浮动体可能出现在实际代码之前。
- **b**　页底（bottom），浮动体被放在一页的底部。
- **p**　独立一页（page），一个或多个浮动体被放在单独的页面中，这个页面被称为浮动页（float page），与之对应，有正文的页面称为文本页（text page）。

例如用选项 [hbp] 就表示允许浮动体出现在环境所在位置、页面底部或单独一页，但不允许出现在一页顶部。浮动体允许位置选项的顺序并不重要，LaTeX 总是以 htbp 的

顺序尝试放置浮动体。不过单独的一个 h 选项通常不总能满足，LaTeX 会把它放宽为 ht 两个可能（参见 7.1.2.4 节），因此，下面的三个浮动体环境开头其实是等效的：

```
\begin{figure}[ht]
\begin{figure}[th]
\begin{figure}[h]
```

如果不设置(允许位置)的参数，figure 和 table 环境默认的位置选项是 tbp。如果图表较多，最好将浮动图表的位置限定设置得宽松一些，以防止浮动体积压过多，最后统一输出。

浮动体最为常见的应用就是直接在 table 环境中放置 tabular 生成的表格，或是在 figure 环境中放置 \includegraphics 命令插入的图形。经常还在前面使用 \centering 命令让图表居中放置，例如：

```
\begin{figure}[htbp] % 允许各个位置
  \centering
  \includegraphics{lion.eps}
\end{figure}
\begin{table} % 默认在页面顶部或单独一页
  \centering
  \begin{tabular}{|c|c|}
    \hline
    图形 & \verb=figure= 环境 \\
    \hline
    表格 & \verb=table= 环境 \\
    \hline
  \end{tabular}
\end{table}
```

5-3-1

不过，无论是 figure 环境还是 table 环境，浮动环境的名称和内容并没有必然联系。一个浮动体只是一个与版心等宽的盒子，内容可以任意放置。figure 环境中也可

图形	figure 环境
表格	table 环境

以是用 TEX 代码或是 ASCII 字符画的图形，table 环境中也可以是以插图形式得到的表格，甚至内容可以和名称毫无干系，如在 figure 环境中放置算法、代码或是很长的公式，也都是可以的。

浮动体的另一个重要功能是使用 \caption 命令加标题，其语法格式如下：

\caption{⟨标题⟩**}**
\caption[⟨短标题⟩**]{**⟨长标题⟩**}**

可选的参数短标题用于图表目录（参见 3.1.1 节），而交叉引用的标签 \label 需要放在 \caption 的后面，或者⟨标题⟩、⟨长标题⟩中。在 \caption 的⟨长标题⟩中可以进行长达多段的叙述，但⟨短标题⟩或单独的 ⟨标题⟩中不允许分段。

一个典型的完整浮动图形可以是这样的：

```
\begin{figure}[htp]
  \centering
  \includegraphics{lion.eps}
  \caption[小狮子]{\TeX{} 系统的吉祥物——小狮子}\label{fig-lion}
% 或作 \caption[小狮子]{\label{fig-lion}\TeX{} 系统的吉祥物——小狮子}
\end{figure}
```

5-3-2

图 5.5　TEX 系统的吉祥物——小狮子

在双栏文档中，figure 和 table 环境就成为只占一栏浮动盒子（宽度是 \columnwidth），其用法与单栏环境中相同。除此之外，LATEX 标准文档类还提供了跨栏排版的图表环境 figure* 和 table*，用来产生跨栏排版的浮动体。跨栏浮动体只允许排在页面的顶部（t）或单独的浮动页面（p）中，其他位置参数会被忽略。figure* 和 table* 环境的默认位置选项都是 tp，在多数情况下，使用 table* 或 figure* 环境的效果就是把内容排在后面一页的顶部。

尽管环境参数中可以设置位置，但浮动体的位置并不总能令人满意，在图表较多时这个问题尤为明显。下面我们来进一步深入探讨 LaTeX 的浮动机制和其他参数。

LaTeX 对每个位置的浮动体的总数和占用大小有一定限制（见表 5.7），超出限制的浮动体会被排在较后的页面中，但可以在 (允许位置) 选项中增加一个 ! 符号，来忽略这些参数限制。因此，在 LaTeX 中最宽松的浮动体位置就是 [!htbp]。在位置选项中加上 ! 号将使浮动体相对更靠近文字或靠前出现。如果使用这一手段浮动体仍然被排在很靠后的位置，就可能需要将浮动体适当前移一段距离了。

与 ! 位置选项相反，\suppressfloats 命令用于禁止浮动体出现在当前页，从而将浮动体的位置向后推。\suppressfloats 命令可以带一个可选参数 t 或 b，表示本页顶部或底部禁止放置浮动体。

当浮动体的排版不符合要求时，可以修改表 5.7 和表 5.8 中的参数。例如，对于含有大量浮动体的文档来说，默认的参数限制都显得过于严格，容易造成浮动体积压无法及时输出的问题，此时就可以修正一些参数，例如：

5-3-3

```
% 放宽浮动体的一些参数
\setcounter{topnumber}{3}
\setcounter{bottomnumber}{2}
\setcounter{totalnumber}{7}
\renewcommand\bottomcaption{0.7}
\renewcommand\textfraction{0.1}
% 严格浮动页的要求
\renewcommand\floatpagefraction{0.7}
```

注意在表 5.7 中的参数相对比较复杂，一些参数还是相互制约的，修改合适的取值可能需要认真的考量和实验，Reckdahl [204, 205] 给出了有关设置这些参数的更详细的建议。

现在我们来更严格地考察 LaTeX 的浮动机制。LaTeX 会把浮动环境做成一个个盒子，所有未输出的浮动体按次序放在队列中处理。LaTeX 所使用的浮动规则可描述如下[136]：

- 浮动体会排在所有位置选项允许的位置中尽可能靠前的地方，但 h 选项优先于 t 选项。亦即，浮动体的位置选项中 h 优先级最高，其后在每一页里按 t, b, p 的优先级排列。

*本节剩余内容初次阅读可略过

表 5.7　限制浮动环境数量和占用大小的参数

参数	类型	默认值*	描述
topnumber	计数器	2	文本页顶部浮动体的最大数量
\topfraction	宏	0.7	文本页顶部浮动体的最大占用空间比例
bottomnumber	计数器	1	文本页底部浮动体的最大数量
\bottomfraction	宏	0.3	文本页底部浮动体的最大占用空间比例
totalnumber	计数器	3	文本页上所有浮动体的最大数量
\textfraction	宏	0.2	文本页中文本所占的最小空间比例
\floatpagefraction	宏	0.5	浮动页中浮动体所占的最小空间比例
dbltopnumber	计数器	2	topnumber 的跨双栏版本
\dbltopfraction	宏	0.7	\topfraction 的跨双栏版本
\dblfloatpagefraction	宏	0.5	\floatpagefraction 的跨双栏版本
\floatsep	弹性长度	12pt plus 2pt minus 2pt	文本页上，处于页顶或页底的多个浮动体之间的垂直间距
\textfloatsep	弹性长度	20pt plus 2pt minus 4pt	文本页上，处于页顶或页底的浮动体与正文之间的垂直间距
\intextsep	弹性长度	12pt plus 2pt minus 2pt	文本页上，使用 h 位置选项排在页面中间的浮动体与上下正文之间的垂直间距
\dblfloatsep	弹性长度	12pt plus 2pt minus 2pt	\floatsep 的跨双栏版本
\dbltextfloatsep	弹性长度	20pt plus 2pt minus 4pt	\textfloatsep 的跨双栏版本

* 标准文档类 10pt 字号的值。

表 5.8 控制浮动页间距的内部命令

参数*	类型	默认值†	描述
\@fptop	弹性长度	0pt plus 1fil	浮动页中页面顶部与浮动体的垂直间距
\@fpsep	弹性长度	8pt plus 2fil	浮动页中多个浮动体之间的垂直间距
\@fpbot	弹性长度	0pt plus 1fil	浮动页中页面底部与浮动体之间的垂直间距

*这几个参数是内部命令，一般只在宏包或文档类中修改。如果在正文导言区进行设置，可以在前后分别加上 \makeatletter 和 \makeatother，参见第 477 页例 8-1-14。
†标准文档类 10pt 字号的值。

- 浮动体不会排在比浮动环境所在处更靠前的页面中。也就是只有在 t 选项生效时，浮动体会排在环境代码位置的同一页，但比浮动环境的上下文略靠前。
- 对于相同类型的浮动环境，多个浮动体会按次序输出。亦即 figure 不会在更早的 figure 之前输出，但可以在更早的 table 之前输出。特别需要注意的是，双栏文档中，跨栏的 figure*、table* 环境和不跨栏的 figure、table 也没有先后制约关系，这可能造成 2 号表格出现在 1 号表格之前，在双栏文档中应该尽量避免混用。
- 只有在浮动体 (*允许位置*) 可选参数中的位置才会放置浮动体。如果省略这个参数，默认的位置参数是 tbp，双栏的跨栏浮动体则是 tp。当仅使用了 h 位置时，LaTeX 会将其扩充为 ht 并发出警告（参见 7.1.2.4 节）。双栏的跨栏浮动体也只有 tp 可以失效。
- 浮动体的排列不能造成页面的上溢出。即浮动体输出时，垂直高度不能超出版心的位置。
- 浮动体的输出必须遵守表 5.7 中的参数限制。不过，如果 (*允许位置*) 参数中有 ! 号，有关文本页的限制将被忽略，只有浮动页的限制（\floatpagefraction 和 \dblfloatpagefraction）起效。

上面的最后三条规则在遇到 \clearpage、\cleardoublepage 或 \end{document} 时会被打破，此时所有队列中未处理的浮动体都会直接输出，(*允许位置*) 的 p 选项也会打开以保证可以将所有浮动体输出。

双栏文档中带星号跨栏的浮动环境与不带星号的环境不能按顺序输出，这无疑是一个 BUG。可以使用 fixltx2e 宏包[164] 来修正这个错误：

5-3-4
```
\usepackage{fixltx2e}
```

fixltx2e 是对 LaTeX 2ε 核心的修正代码，除了双栏浮动体还做了其他一些改进，但因为考虑兼容性问题才没有直接修改内核。除了它以外，也可以加载 dblfloatfix 宏包[108]

修正类似的问题，这个宏包专门用来修正双栏浮动体的问题，同时还允许跨栏浮动体的 b 选项生效，使浮动环境的位置更为灵活，不过不要同时使用这两个宏包。

LaTeX 最多保存 18 个未处理的浮动体。如果有过多的浮动体积压不能及时输出，就会报错（参见 7.1.2.2 节）。通常遇到这种问题只需要增加浮动体允许输出的位置，或是用 \clearpage 立即输出所有的浮动体。不过，如果确实要求保留原来参数中设定的位置，则可以使用 morefloats[107] 宏包增大保存未处理浮动体的限制。

另外需要指出的是，脚注和边注（参见 2.2.7 节）也是特殊的浮动体，它们的输出位置与普通的浮动体不同，但同样具有浮动的效果（太长而排不下的脚注和边注会排在后面的页面），会占用未处理的浮动体个数。

5.3.2　标题控制与 caption 宏包

在设计文档时，我们经常需要修改浮动标题的字体、间距、对齐方式等格式，然而 LaTeX 2ε 内核及标准文档类并没有提供直接修改浮动标题格式的命令，这时就可以使用 caption 宏包来完成相关的设置。

注意，在这一小节的示例中，浮动体标题显示的是采用 LaTeX 2ε 默认的格式。为了方便，许多例子省略了外面的 figure、table 等浮动环境，使用“浮动体”代替“图”、“表”或“Figure”、“Table”字样，但在实际的文档中并不能随意略去外面的浮动环境。

使用 caption 宏包设置标题格式是通过一系列键值对形式的选项完成的，这些选项既可以作为宏包的可选项，也可以作为 \captionsetup 命令的参数出现。例如，如果要让 \caption 标题的字体缩小一号，同时数字标签使用粗体，就可以用：

```
\usepackage[font=small,labelfont=bf]{caption}
```

5-3-5

来完成，也可以改为：

```
\usepackage{caption}
\captionsetup{font=small,labelfont=bf}
```

5-3-6

还可以利用 \captionsetup 的可选参数，只修改 figure 环境的标题格式：

```
\usepackage{caption}
\captionsetup[figure]{font=small,labelfont=bf}
```

5-3-7

*本节内容初次阅读可略过。

这正是 caption 宏包的基本用法。

也可以使用 \captionsetup 命令单独设置某种浮动体的格式，只要添加一个可选参数说明要修改的浮动环境，例如：

5-3-8
```
\usepackage{caption}
\captionsetup[figure]{font=small,labelfont=bf}
```

caption 宏包提供了非常多的选项，限于篇幅，这里只选择其中常用的一些。为了叙述方便，在本节后面的例子中，我们不使用实际的 figure 或 table 环境，而是假定 \caption 放在了一个虚拟的 metafloat 浮动环境中，这个浮动环境的标题是“浮动体”，输出格式也基于标准文档类下的默认的浮动标题格式。

首先是 format 选项，caption 宏包预定义了 plain 和 hang 两种格式，默认的格式是 plain，例如：

5-3-9
```
\caption{默认居中的短标题}
```

浮动体 1: 默认居中的短标题

5-3-10
```
\caption[plain 格式]{plain 格式下，
如果标题很长，折成几行，就会像普通的
正文段落一样显示。

只要设置好前面的短标题，就可以把浮动
标题分成好几段。}
```

浮动体 2: plain 格式下，如果标题很长，折成几行，就会像普通的正文段落一样显示。
只要设置好前面的短标题，就可以把浮动标题分成好几段。

5-3-11
```
\captionsetup{format=hang}
\caption[hang 格式]{hang 格式的效果
是，对于很长的标题，前面的数字标签会
进行悬挂缩进，就好像 \LaTeX{} 的列表
环境一样。}
```

浮动体 3: hang 格式的效果是，对于很长的标题，前面的数字标签会进行悬挂缩进，就好像 LaTeX 的列表环境一样。

labelformat 选项则用来设置标签编号的格式，例如：

5-3-12
```
% \captionsetup{labelformat=default}
\caption{默认格式，同 simple}
```

浮动体 4: 默认格式，同 simple

```
\captionsetup{labelformat=empty}
\caption{空格式}
```

空格式　5-3-13

```
\captionsetup{labelformat=simple}
\caption{简单格式，直接输出}
```

浮动体 6: 简单格式，直接输出　5-3-14

```
\captionsetup{labelformat=brace}
\caption{数字右括号，无文字}
```

浮动体 7): 数字右括号，无文字　5-3-15

```
\captionsetup{labelformat=parens}
\caption{带括号数字，无文字}
```

浮动体 (8): 带括号数字，无文字　5-3-16

labelsep 选项控制标签与后面标题之间的间隔。

```
\captionsetup{labelsep=none}
\caption{没间隔}
```

浮动体 9没间隔　5-3-17

```
% \captionsetup{labelsep=colon}
\caption{英文分号（默认）}
```

浮动体 10: 英文分号（默认）　5-3-18

```
\captionsetup{labelsep=period}
\caption{英文句点}
```

浮动体 11. 英文句点　5-3-19

```
\captionsetup{labelsep=space}
\caption{空格}
```

浮动体 12 空格　5-3-20

```
\captionsetup{labelsep=quad}
\caption{一个 em 的间隔}
```

浮动体 13　一个 em 的间隔　5-3-21

```
\captionsetup{labelsep=newline}
\caption{标题会另起一行}
```

浮动体 14
标题会另起一行　5-3-22

5-3-23

```
\captionsetup{labelsep=endash}
\caption{en dash 连接符}
```

浮动体 15 – en dash 连接符

justification 选项设置浮动标题的对齐方式，可用的选项值见表 5.9。默认情况下，单行的标题居中对齐，而多行的标题则与普通段落一样均匀对齐。

表 5.9 caption 宏包的 justification 选项

选项值	对应段落命令	对齐方式
justified	无（\justifying*）	普通段落的均匀对齐，默认值
centering	\centering	每行居中对齐
centerlast	\centerlast†	每段的最后一行居中对齐，其他行均匀对齐
centerfirst	\centerfirst†	仅标题第一行居中，其他行均匀对齐
raggedright	\raggedright	每行左对齐，段落右边界可以不对齐
RaggedRight	\RaggedRight*	改进的 raggedright
raggedleft	\raggedleft	每行右对齐

*需要 ragged2e 宏包。

†由 caption 宏包单独提供。

与标准文档类一样，在默认情况下，caption 宏包在单行的短标题中会忽略 justification 选项，而将其居中排版，只有多行的标题才使用选项中的对齐方式。但如果希望设置的对齐方式对单行的标准也有效，则可以使用 singlelinecheck=false 来关闭对单独一行标题的检测。例如，要得到总是右对齐的标题，就可以用：

5-3-24

```
\captionsetup{
  justification=raggedleft,
  singlelinecheck=false}
\caption{右对齐的标题}
```

浮动体 16: 右对齐的标题

font 用来设置浮动标题的字体，而 labelfont 和 textfont 则可以单独设置前面的标签和后面文字的字体。可以使用的字体选项包括字号大小、字体族、字体系列、字体形状、行距、文字颜色等。所有可用的选项值见表 5.10。

多个不同的字体选项可以同时使用，只要把几个选项放在分组中，例如：

表 5.10　caption 宏包预定义的 font 选项值

类别	选项值	等价字体命令	效果
字号	scriptsize	\scriptsize	非常小
	footnotesize	\footnotesize	很小
	small	\small	较小
	normalsize	\normalsize	正文文字大小
	large	\large	较大
	Large	\Large	很大
字体族	rm	\rmfamily	罗马体 Roman family
	sf	\sffamily	无衬线体 Sans Serif family
	tt	\ttfamily	打字机体 Typewriter family
字体系列	md	\mdseries	中等粗细 Medium series
	bf	\bfseries	**粗体 Bold series**
字体形状	up	\upshape	直立体 Upright shape
	it	\itshape	意大利体 *Italic shape*
	sl	\slshape	倾斜体 *Slanted shape*
	sc	\scshape	小型大写字母 SMALL CAPS SHAPE
行距 †	singlespacing	\singlespacing	单倍行距
	onehalfspacing	\onehalfspacing	“1.5 倍”行距
	doublespacing	\doublespacing	“双倍”行距
	stretch=⟨倍数⟩	\setstretch{⟨倍数⟩}	多倍行距
颜色	normalcolor	\normalcolor	默认颜色
	color=⟨颜色⟩	\color{⟨颜色⟩}	指定彩色
选项集合	normalfont	\normalfont	恢复默认字体
	normal		恢复默认字体、行距、颜色

†这里的行距设置使用 setspace 宏包的命令，其中 1.5 倍和双倍行距的意义可能与其他地方不同，参见 2.1.4 节。

5-3-25
```
\captionsetup{font={small,sf},
              labelfont=bf}
\caption{小号加粗无衬线体 Caption}
```

浮动体 17: 小号加粗无衬线体 Caption

几个字体选项还支持 += 的语法，用于在现有设置的基础上增加新设置，如：

5-3-26
```
\captionsetup{font=small}
\captionsetup{font+=bf}
\caption{小号加粗字体 Caption}
```

浮动体 18: 小号加粗字体 Caption

margin 选项用来设置标题距离页面左右边界的距离，width 则用来设置标题的最大宽度。这两个选项有制约关系，因而通常同时只使用其中一个，例如：

5-3-27
```
\captionsetup{margin=4em}
\caption{标题距离左右各 4\,em
  的距离。}
```

浮动体 19: 标题距离左右各 4 em 的距离。

5-3-28
```
\captionsetup{width=6em}
\caption{标题最多只有 8\,em 宽。}
```

浮动体 20: 标题最多只有 8 em 宽。

skip 选项控制标题与浮动环境内容的垂直间距，在标准文档类中的默认值是 10 pt，例如：

5-3-29
```
\captionsetup{skip=0pt}
浮动体的内容。
\caption{与前面无额外间距。}
```

浮动体的内容。
浮动体 21: 与前面无额外间距。

type 选项可以设置标题所对应的浮动环境类型，这就允许在非浮动环境中直接使用浮动体的标题，或者是在同一个浮动体里面显示不同的几个标签。但注意标题仍然应该被放在一个环境中或盒子中，而不要直接写在正文里面。例如，可以利用 type 在同一个浮动体中完成图表的混排：

```
\begin{figure}
```

```
\begin{minipage}[b]{.5\textwidth}
\centering
\includegraphics[width=.4\textwidth]{texlive-lion.pdf}
\caption{\TeX\ Live 吉祥物狮子}
\end{minipage}%
\begin{minipage}[b]{.5\textwidth}
\centering
\begin{tabular}{|*{5}{c|}}
  \hline
  1996 & 1998 & 1999 & 2000 & 2001 \\ \hline
  2002 & 2003 & 2004 & 2005 & 2007 \\ \hline
  2008 & 2009 & 2010 & \dots & \\
  \hline
\end{tabular}
\captionsetup{type=table}
\caption{\TeX\ Live 的版本}
\end{minipage}
\end{figure}
```

5-3-30

图 5.6: TEX Live 吉祥物狮子

1996	1998	1999	2000	2001
2002	2003	2004	2005	2007
2008	2009	2010	…	

表 5.11: TEX Live 的版本

在导言区使用 某name 选项可以用来设置标题标签的文字名称。在标准文档类中，figure 与 table 环境的名称是 Figure 和 Table，而 ctex 文档类则分别是“图”和“表”，设置 figurename 和 tablename 选项等价于修改宏 \figurename 或是 \tablename 的值，但更为方便，例如：

```
% 导言区
\usepackage{caption}
\captionsetup{figurename=图片}
```

5-3-31

直接使用 name 选项则可以在浮动体环境中临时性修改标签名称，例如：

5-3-32
```
\captionsetup{
  type=figure,name=空图片}
\caption{标签名称可以修改}
```

空图片 5.7: 标签名称可以修改

caption 宏包除了定义了大量的格式选项，同时也额外提供了一些有用的命令。例如，\captionof{⟨类型⟩}{⟨标题⟩} 命令可以看做是先设置了 type 选项，然后使用普通的 \caption：

5-3-33
```
\captionof{figure}{空图片标题}
```

图 5.8: 空图片标题

\caption 和 \captionof 都有一个带星号 * 的形式，表示一个不编号、不显示标签也不进入图表目录的标题，例如：

5-3-34
```
\captionsetup{font=sf}
\caption*{无编号的标题，只保留格式}
```

无编号的标题，只保留格式

\ContinuedFloat 命令则用来放在浮动体中，阻止标题的编号增加，从而可以用一个编号表示多个浮动体。如要产生“续图”、“续表”的功能，就可以使用类似下面的代码：

5-3-35
```
\begin{figure}
\ContinuedFloat
  ......
\caption{某图形}
\end{figure}
\begin{figure}
\ContinuedFloat
  ......
\caption{某图形（续）}
\end{figure}
```

最后，caption 宏包同时也提供了许多命令来为其格式选项增加新的取值。相关的命令很多，作为例子，我们来看看如何为 labelsep 选项声明一个 fullcolon 的取值，其效果是使用全角的冒号来分隔标签和标题文字：

```
% 一般在导言区使用
\DeclareCaptionLabelSeparator{fullcolon}{：} % 声明中文的全角冒号分隔符
\captionsetup{labelsep=fullcolon} % 为中文的标题设置全角冒号分隔符
```

5-3-36

bicaption[240] 是 caption 宏包的附加宏包，它提供了双语标题的功能，其基本命令是 \bicaption，语法格式如下：

\bicaption[⟨短标题一⟩]{⟨长标题一⟩}[⟨短标题二⟩]{⟨长标题二⟩}

同时可以使用 \captionsetup[bi-first] 与 \captionsetup[bi-second] 的 lang 选项分别设置两个标题不同的语言。bicaption 原本使用 babel 宏包或 polyglossia 宏包提供的语言选择机制来设置不同语言的标题，不过中文等东亚语言不使用上述宏包的翻译机制，就需要手工设置不同语言的标题。我们可以使用 \DeclareCaptionOption 命令来声明一个新的选项，完成标签名的重定义，然后可以用 \captionsetup 为每种语言分别调用。例如，要设置中英文两种图表标题，可以在导言区使用：

```
% 中文文档类会设定好标题的第一种语言
\documentclass{ctexart}
\usepackage{bicaption}
% 声明 english 选项重定义第二种语言的标签名，选项没有参数
\DeclareCaptionOption{english}[]{%
  \renewcommand\figurename{Figure}%
  \renewcommand\tablename{Table}}
\captionsetup[bi-second]{english}
```

5-3-37

之后就可以在正文中得到双标题的浮动体了：

```
\begin{figure}
  \centering FIGURE
  \bicaption{中文标题}{English Title}
\end{figure}
```

FIGURE

图 5.9: 中文标题

Figure 5.9: English Title

5-3-38

练 习

5.9 查看 caption 宏包手册 Sommerfeldt [238]，如果需要把标题的中文字体改为隶书，应该如何设置？

5.10 查看宏包手册 Sommerfeldt [238]，看看 caption 宏包的 hypcap 参数有什么作用。

caption 还是 caption2？

在一些文档中，如 [204]，往往会看到使用 caption2 代替 caption 宏包的建议。然而我们这里仍然使用的是 caption 宏包，这又是为什么呢？

事实上，这涉及到宏包的版本更迭问题。caption 与 caption2 其实是同一个宏包的不同版本，最早的版本就叫做 caption，版本号按 1.*x* 变化，然而第一版 caption 在功能上有很多缺陷，作者 Sommerfeldt 推出了实验性质的第二版，版本号按 2.*x* 变化，也就是 caption2。之所以取不同的名字很大程度上是因为两个版本的宏包虽然在功能上类似，但语法结构已经有很大的区别，新旧宏包并不兼容。

今天我们在这里使用的则是 caption 宏包的第三个版本，版本号按 3.*x* 变化。新版本的 caption 宏包与前面两个版本的语法又发生了许多变化，两套旧版本都不支持键值式的语法，而现在的版本则使用键值对给出了更为清晰一致的设置方式。确实有一个 caption3.sty，它现在其实是 caption 的内核，提供了 caption 宏包的基本功能，而我们直接使用的 caption.sty 则提供了宏包选项以及与为兼容其他宏包而编写的大量代码。历史绕了一个圈子，宏包的名称又回到了起点。

这类旧版本被新版本取代，同时名称和使用方式都发生变化的宏包并不在少数。在 3.4.3.2 节介绍的 glossaries 宏包，它的旧版本就叫做 glossary，并且已经从一些新的 TeX 发行版中删掉了。为此，glossaries 宏包还专门附带了一个从旧版本迁移向新版本的手册 Talbot [251]，帮助老用户转向新的版本上来。另一类情况是有其他人对旧宏包做了改进，另起一个名字发布的，5.1.4 节介绍的 xtab 宏包，就是对 supertabular 改进得到的，我们也应该尽量使用改进的新版本。甚至 LaTeX 内核本身也有这个问题，已被废弃的旧版本是 LaTeX 2.09，而现在的版本则是 LaTeX 2$_\varepsilon$，二者并不完全兼容。因而在查看资料或使用宏包时，也应该查看本机上的手册，留意当前版本的信息，以免出现问题。

尽管当年高德纳教授设计 TeX 的目的之一就是保证文档稳定和代码的向前兼容，但大量第三方宏包的出现已经渐渐打破了这一初衷。当我们享受不断进步的宏包所带来的便利时，也不得不接受 LaTeX 代码也可能过时这一事实。同样，本书的一些内容也将无可避免地在未来变得不符合实际，对此我们只能提前做好准备。

5.3.3　并排与子图表

在实际中，经常需要把好几个图表并列放在一起输出。由于 LaTeX 的浮动环境并不对环境内容加以限制，所以只要直接把图表放在一个浮动体里面就可以了，例如：

```
\begin{table}
  \centering
  \caption{并排的表格}
  \begin{tabular}{|c|c|}
    \hline 图 & 表 \\ \hline
  \end{tabular}%
  \qquad
  \begin{tabular}{|c|c|}
    \hline Figure & Table \\ \hline A & B \\ \hline
  \end{tabular}
\end{table}
```

5-3-39

表 5.12　并排的表格

图	表

Figure	Table
A	B

tabular 环境生成的表格和 \includegraphics 插入的图形都是一个大的盒子，因而可以直接并排放在一起。不过如果是和一段文字并排放在一起，则可以使用 \parbox 或 minipage 环境生成一个子段盒子：

```
\begin{figure}
  \centering
  \includegraphics[width=0.4\textwidth]{texlive-lion.pdf}%
  \qquad
  \parbox[b]{0.4\textwidth}{这只狮子是由画师 Duane Bibby 专门为著名的
  \TeX{} 发行版 \TeXLive{} 绘制的作品。狮子是 \TeX{} 系统的吉祥物，
  Duane Bibby 创作了大量有关 \TeX{} 狮子的插图，如高德纳的 \textit{The
```

*本节内容初次阅读可略过。

```
  \TeX{}book} 与 Lamport 的 \textit{\LaTeX: A Document Preparation
  System} 两书中的狮子插图，就是由 Duane Bibby 创作的。}
  \caption{\TeX{} 狮子}\label{fig:texlivelion}
\end{figure}
```

5-3-40

这只狮子是由画师 Duane Bibby 专门为著名的 TEX 发行版 TEX Live 绘制的作品。狮子是 TEX 系统的吉祥物，Duane Bibby 创作了大量有关 TEX 狮子的插图，如高德纳的 *The TEXbook* 与 Lamport 的 *LATEX: A Document Preparation System* 两书中的狮子插图，就是由 Duane Bibby 创作的。

图 5.10 TEX 狮子

在使用并排的图表或文字时，需要注意其对齐方式。tabular 环境和 \parbox 生成的子段盒子，默认都是在盒子中央对齐（盒子的基准点是中线左端），而 \includegraphics 插入的图形其基准点则在左下角，因此图 5.10 就使用了 \parbox 的 b 选项使文字与前面的图形对齐。如果需要让插图垂直居中对齐，则可以把它放进子段盒子中。

然而要让两个插图按顶部对齐，就需要一点技巧。注意如果直接把插图放进 t 选项的子段盒子中，并不能使图形在顶部对齐，因为 t 选项只能让第一行按基线对齐。此时可以在盒子里面先使用 \vspace{0pt} 增加一个高度为 0 的空行，然后按这个空行对齐。图形的宽度可能不能直接确定，可以用 varwidth 宏包的 varwidth 处理：

```
\begin{figure}
  \centering
  \begin{varwidth}[t]{\textwidth}
    \vspace{0pt}
    \includegraphics{lion.eps}
  \end{varwidth}%
  \qquad
  \begin{varwidth}[t]{\textwidth}
```

```
    \vspace{0pt}
    \includegraphics[height=4cm]{texlive-lion.pdf}
  \end{varwidth}
  \caption{两幅狮子图形的按顶部对齐}
\end{figure}
```

5-3-41

图 5.11　两幅狮子图形的按顶部对齐

在一个浮动环境中可以有多个 \caption 标题[①]，甚至可以利用 caption 宏包的 type 选项同时使用不同类型的标题，例如：

```
\begin{table}
\parbox[b]{.5\textwidth}{\centering
  \caption{文字表格}
  \begin{tabular}{|c|c|}
    \hline 图 & 表 \\ \hline
  \end{tabular}}%
\parbox[b]{.5\textwidth}{\centering
  \caption{数学表格}
  $\begin{array}{|c|c|}
    \hline \sqrt{2} & 1.414\dots \\ \hline
    \sqrt{3} & 1.732\dots \\ \hline
  \end{array}$}
\end{table}
```

5-3-42

① 但如果是使用 float 宏包的 \newfloat 自定义的浮动体，或是用 \restylefloat 命令对 figure, table 环境做了重定义，则只能使用一个 \caption 标题，而用 caption 宏包的 \DeclareCaptionType 定义的新类型则没有问题，参见 5.3.4 节。

表 5.13 文字表格

图	表

表 5.14 数学表格

$\sqrt{2}$	1.414...
$\sqrt{3}$	1.732...

有时候我们还需要更复杂的子图表功能：可以给整个浮动体加一个概括性的标题，同时对浮动体内的每个子图表，也都有自己的编号和标题。这可以使用 caption 的一个附加宏包 subcaption 宏包[239] 来完成。subcaption 依赖 caption 宏包，使用时需要同时调用两个。

subcaption 宏包提供了一组命令来完成子图表的排版输出。\subcaption 命令用来直接输出子标题，例如，我们将例 5-3-42 中的两个 \caption 换成 \subcaption，同时加上整体的标题，则有：

5-3-43

```
% \usepackage{caption,subcaption}
\begin{table}
\caption{图表的子标题}
\parbox[b]{.5\textwidth}{\centering
  \begin{tabular}{|c|c|}
    \hline 图 & 表 \\ \hline
  \end{tabular}
  \subcaption{文字表格}}%
\parbox[b]{.5\textwidth}{\centering
  $\begin{array}{|c|c|}
    \hline \sqrt{2} & 1.414\dots \\ \hline
    \sqrt{3} & 1.732\dots \\ \hline
  \end{array}$
  \subcaption{数学表格}}
\end{table}
```

表 5.15 图表的子标题

图	表

(a) 文字表格

$\sqrt{2}$	1.414...
$\sqrt{3}$	1.732...

(b) 数学表格

由于子图表几乎总是需要使用子段盒子来放置内容和子标题，所以 subcaption 宏包还同时提供了 subfigure 和 subtable 环境，它们的语法和功能与 minipage 完全相同，

只是在里面可以直接使用 \caption 命令来表示子标题，如：

```
% \usepackage{caption,subcaption}
\begin{table}
  \caption{子图表环境}
  \begin{subtable}[b]{.5\textwidth}
    \centering
    \begin{tabular}{|c|c|} \hline 图 & 表 \\ \hline \end{tabular}
    \caption{文字表格}
  \end{subtable}%
  \begin{subtable}[b]{.5\textwidth}
    \centering
    $\begin{array}{|c|c|}
      \hline \sqrt{2} & 1.414\dots \\ \hline
      \sqrt{3} & 1.732\dots \\ \hline
    \end{array}$
    \caption{数学表格}
  \end{subtable}
\end{table}
```

5-3-44

表 5.16　子图表环境

图	表

(a) 文字表格

$\sqrt{2}$	1.414...
$\sqrt{3}$	1.732...

(b) 数学表格

使用更为方便的则是 \subcaptionbox 与 \subcaptionbox* 命令，能生成一个带有子标题的子图表例子（带 * 的命令不编号），其语法格式如下：

\subcaptionbox[⟨目录标题⟩**]{**⟨标题⟩**}[**⟨宽度⟩**][**⟨盒子内位置⟩**]{**⟨内容⟩**}**
\subcaptionbox*{⟨标题⟩**}[**⟨宽度⟩**][**⟨盒子内位置⟩**]{**⟨内容⟩**}**

其中 ⟨宽度⟩ 变成可选的参数，如果省略就使用其 ⟨内容⟩ 的自然宽度，类似使用了 varwidth 环境的效果；⟨盒子内位置⟩ 确定 ⟨内容⟩ 在盒子中的水平对齐方式，可以是 l（\raggedright）、r（\raggedleft）、c（\centering）或 s（无特别格式），默认为居中的 c。例如：

```
% \usepackage{caption,subcaption}
\begin{table}
\caption{子图表盒子}
\centering
\subcaptionbox{文字表格\label{subtab:test}}[6em]{%
  \begin{tabular}{|c|c|} \hline 图 & 表 \\ \hline \end{tabular}}\qquad
\subcaptionbox{数学表格}{%
  $\begin{array}{|c|c|}
    \hline \sqrt{2} & 1.414\dots \\ \hline
    \sqrt{3} & 1.732\dots \\ \hline
  \end{array}$}
\end{table}
```

5-3-45

表 5.17 子图表盒子

(a) 文字表格

图	表

(b) 数学表格

$\sqrt{2}$	1.414...
$\sqrt{3}$	1.732...

如例 5-3-45 所示，使用 \subcaptionbox 时，需要给子图加引用的 \label 标签可以放在 ⟨标题⟩ 参数中。使用 \ref 引用标签 subtab:test 将得到“5.17(a)”，它是外层与内层编号的混合。若只引用子标题的内层编号，可以用 subcaption 提供的 \subref 命令，如 \subref{subtab:test} 将得到“(a)”。

与 caption 宏包相同，子标题的格式仍然可以使用 subcaption 的宏包选项全局设置，或利用 \captionsetup 命令全局或局部地进行设置，如：

```
\usepackage{caption}
\usepackage{subcaption}
\captionsetup[sub]{font={small,it}} % 设置所有子标题
\captionsetup[subtable]{labelformat=simple,labelsep=colon} % 设置子表格
```

5-3-46

除了 subcaption 宏包，也可以使用 subfig 宏包[55] 和 floatrow 宏包[140] 宏包来排版子图表。它们都与 caption 宏包的功能兼容，同时提供额外的子图表排版功能。subfig 主要提供了 \subfloat 和 \subref 命令，功能和语法都与 subcaption 十分相近[①]。floatrow

① subfig 是 Cochran 在其旧版本的 subfigure 的基础上编写的宏包，旧版本与过时的 caption2, ccaption 等宏包兼容，新版本则基于第 3 版的 caption 宏包。在 Reckdahl [204] 等较早的资料中介绍的主要是 subfigure 宏包，而新的文档多介绍

宏包则预定义了许多更为复杂的子图表格式，这里就不多做介绍了。

5.3.4　浮动控制与 float 宏包

关于浮动体，提出最多的一个问题就是：怎样让图表不要乱跑？习惯于所见即所得环境下拖曳鼠标放置图形的人尤其不适应浮动环境的“奇怪”效果。浮动图表的目的是用浮动的位置来避免糟糕的分页，但如果不在乎因为图表太大而产生的分页，而要求有确定的位置，那么这其实是要求不使用“浮动”环境。把图表简单地放在 center、quote 等环境中就可以做到这一点，5.3.2 节还展示了如何在这种情况下仍然使用图表的标题。

无论如何，float 宏包[149] 还是为标准的浮动环境提供了一个新的 H 位置选项用来产生没有浮动效果的图表环境，它的使用非常简单，和一般的浮动环境没有什么区别：

```
% \usepackage{float}
\begin{figure}[H]
\centering
\includegraphics[height=1cm]{lion.eps}
\caption{不浮动的图表}
\end{figure}
```

图 5.12　不浮动的图表

5-3-47

H 选项不能与 h, t, b, p 等其他位置选项混用。H 表示 Here，事实上使用了 H 选项的 figure 或 table 环境就不再是一个浮动体，而只是一个前后间距与内容格式都与普通浮动环境相同的一个大盒子。float 宏包提供的 H 选项比用 center 环境和 caption 宏包的 \captionof 命令来模拟普通浮动环境的格式要更准确，也更自然一些，这个选项也是 float 宏包的主要功能。

除此之外，float 宏包还提供了定义新浮动环境的功能，这是由 \newfloat 命令完成的，其语法格式如下：

\newfloat{⟨环境名⟩}{⟨位置⟩}{⟨目录文件扩展名⟩}[⟨上级计数器⟩]

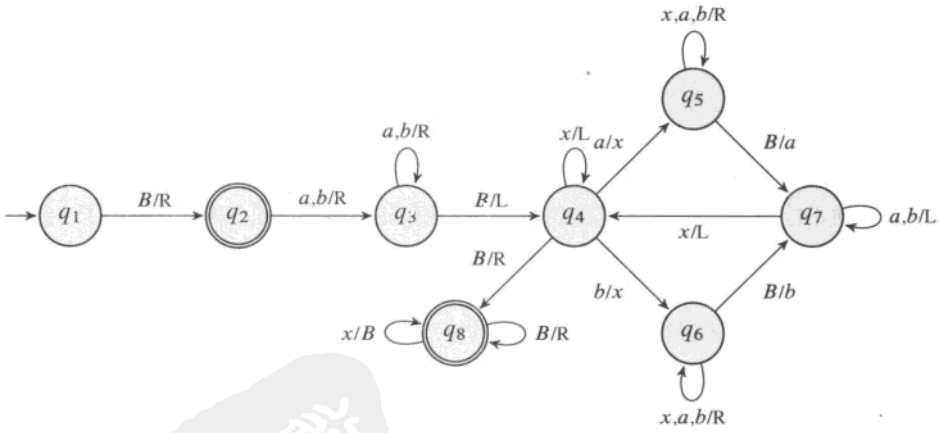
subfig。不过 subfig 的功能与 caption 自带的 subcaption 大体重复，手册和实现代码也更冗长，因而选用 subcaption 更为合适。

*本节内容初次阅读可略过。

其中〈位置〉是浮动环境默认的位置选项，可以使用h, t, b, p或单独的H。〈目录文件扩展名〉是用于产生与图表目录（参见 3.1.1 节）类似的目录的辅助文件扩展名，而可选的〈上级计数器〉则可以让浮动环境按章节编号。使用 \newfloat 定义了新的浮动体后，一般还要用 \floatname 命令定义这个浮动体的标题标签名，例如，可以为文档中的流程图单独定义一个环境：

```
% 导言区
% \usepackage{float}
\newfloat{flowchart}{htbp}{loflow}[chapter]
\floatname{flowchart}{流程图}
% 正文
\begin{flowchart}
  \centering
  \includegraphics{turing-reverse.pdf}
  \caption{求逆字符串的图灵机}
\end{flowchart}
```

5-3-48



流程图 5.1 求逆字符串的图灵机

float 宏包还提供了 \floatstyle{〈格式〉} 命令，可以使之后用 \newfloat 定义的所有浮动体按指定的格式输出，也可以在后面使用 \restylefloat{〈环境名〉} 来指定原有浮动环境的格式，可选的格式如下：

- **plain** 这是默认的格式，它与标准文档类 figure, table 环境的格式基本相同，只是 \caption 产生的标题总在浮动环境的底部。
- **plaintop** 标题在顶部，其余与 plain 格式相同。
- **boxed** 浮动体内容在一个线框中，标题在线框下面。
- **ruled** 浮动体类似三线表的格式，标题在顶部，标题前后与浮动体后面各有一条横线。Graham et al. [87] 一书的图表就是这种样式。

这里举一个 ruled 格式的例子：

```
\floatstyle{ruled}
\restylefloat{flowchart}
% 正文
\begin{flowchart}
  \fbox{A} $\longrightarrow$ \fbox{B} \\
  \fbox{C} $\longrightarrow$ \fbox{D}
\caption{基本流程图}
\end{flowchart}
```

5-3-49

流程图 5.2 基本流程图

A ⟶ B
C ⟶ D

float 宏包的 \floatplacement 则可以重定义浮动环境的默认位置参数，例如让图表环境都允许 h 位置：

```
\floatplacement{figure}{htbp}
\floatplacement{table}{htbp}
```

5-3-50

而 float 的 \listof{⟨环境名⟩}{⟨标题⟩} 命令的功能则与命令 \listoffigures 和 \listoftables 类似，用来输出新定义的浮动环境的标题目录，例如：

```
\listof{flowchart}{流程图目录}
```

5-3-51

需要注意的是，使用 float 的 \newfloat 或 \refstyle 重定义浮动环境中只能使用一个 \caption 标题，标题的位置也固定为顶部或底部，这对于排版并列图表非常不便。caption 宏包的个别功能也因此受到影响，rule 格式中标题格式则完全固定。因此，如果对标题有较高要求，更好的方式是使用 newfloat 宏包[241] 的 \DeclareFloatingEnvironment 命令来定义新的浮动体，其语法格式如下：

\DeclareFloatingEnvironment[⟨选项表⟩]{⟨环境名⟩}

其中 ⟨选项表⟩ 可以使用下面的选项：

- name=⟨标签名⟩
- listname=⟨目录名⟩
- fileext=⟨目录文件扩展名⟩，默认是 lo⟨环境名⟩
- placement=⟨位置参数⟩
- within=⟨上级计数器⟩，可以为 none（无）
- chapterlistsgaps= on 或 off，开关在目录中，不同章浮动体标题间的额外间距。

因此，定义流程图的格式也可以使用如下命令：

```
\usepackage{newfloat}
\DeclareFloatingEnvironment[fileext=loflow, placement=htbp,
  within=chapter, name=流程图, listname=流程图目录]{flowchart}
```

使用这种方式定义的新浮动环境没有标题方面的限制，可以与 caption 更好的结合。可以使用 \listof⟨环境名⟩s 这样的命令来输出目录，如前面定义的流程图就使用 \listofflowcharts。

除了 float 宏包及其 H 选项，还有一些工具可以用来控制浮动体的位置。

placeins 宏包[13] 提供了一个 \FloatBarrier 命令，顾名思义，它会在所在的位置产生一个无形的屏障，所有之前的浮动体都必须在这个屏障之前输出。\FloatBarrier 命令的功能有些像 \clearpage，不过只要可能，\FloatBarrier 命令会避免直接分页。placeins 宏包有几个选项，section 选项会在每个 \section 命令之前隐含地增加一个 \FloatBarrier 命令，使浮动体局限在一节的范围之内；above 和 below 选项可以放宽 \FloatBarrier 命令的位置限制，使浮动体可以出现在同一页的较前或较后的位置。

float 的 H 选项给出了确切的“此处”位置，也可以使用 afterpage 宏包[42] 来得到确切的“下一页顶部”位置。afterpage 宏包提供了一个 \afterpage 命令，可以把参数中的内容放在下一页的开头，同时不影响正常的正文流向。因此，可以使用

5-3-52

```
% \usepackage{afterpage}
\afterpage{\begin{figure}[H]
  ...
\end{figure}}
```

来得到下一页顶部的浮动图形环境，而如果使用

```
% \usepackage{afterpage}
\afterpage{\clearpge}
```

5-3-53

则会产生与 placeins 的 \FloatBarrier 有些类似的效果，它强制所有浮动体在下一页之前输出完毕（其中的 \clearpage 也可以改成 \FloatBarrier）。

有的期刊会要求稿件将所有的图表放在整个文章的末尾，但把整个浮动环境挪到文档末尾却非常不方便修改，endfloat 宏包[156] 就解决了这个问题。引用 endfloat 宏包后，所有的浮动图表都会被放在文档最后。也可以使用 \processdelayedfloats 命令直接输出之前延迟输出的浮动图表，例如放在文档正文之后，参考文献列表之前。更多的命令的选项控制，可参见文档 McCauley and Goldberg [156]。

5.3.5　文字绕排

让文字沿图表绕排一直困扰着不少使用 LaTeX 的人。对于小幅的图表，使用绕排的方式可以得到更为紧凑的页面，在篇幅紧张或注重行文的场合，效果往往比浮动环境更有吸引力。这一节主要将介绍两个有关文字绕排的宏包：picinpar 宏包[242] 和 wrapfig 宏包[14]。由于 TeX 固有的限制，文字绕排的效果还无法做到尽善尽美，对绕排图表的位置、形状、使用都有一些限制。因此在使用绕排工具时，往往需要仔细的调整，或者另寻他途。

picinpar 提供了 figwindow 和 tabwindow 环境来实现绕排功能。它们的语法格式如下（中括号中的 4 个参数都是必选的）：

```
\begin{figwindow}[⟨下降行数⟩,⟨水平位置⟩,⟨图内容⟩,⟨图标题⟩]
    ⟨绕排文字⟩
\end{figwindow}

\begin{tabwindow}[⟨下降行数⟩,⟨水平位置⟩,⟨表内容⟩,⟨表标题⟩]
    ⟨绕排文字⟩
\end{tabwindow}
```

figwindow 和 tabwindow 环境会在 ⟨绕排文字⟩ 的段落中开一个窗口，用来放置图表。图表的位置由前两个参数确定，⟨下降行数⟩ 是一个整数，确定被绕排图表的垂直位

*本节内容初次阅读可略过。

置，图表将在这么多行文字下方显示；(水平位置) 可以是 l, c 或 r，表示窗口开在段落左、中、右的位置。后面两个参数分别是图表的内容与标题。标题可以留空，但需要保留标题前的逗号，此时就没有标题和编号。如果标题的编号需要引用，可以把标签放在标题内。下面是一个使用 figwindow 的例子[①]：

5-3-54

```
% \usepackage{picinpar}
\begin{figwindow}[2,c,% 跨过段落的前两行，中间位置
  \includegraphics{lion.eps},Lion\label{fig:wraplion}]
\lipsum*[1] % 足够长的文本段落
\end{figwindow}
```

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

图 5.14 Lion

picinpar 是一个 LaTeX 2.09 时代的宏包，不过至今仍能正常使用。它能自动计算图表内容的大小，在环境中文本段落指定的位置开窗口放置此图表。不过，picinpar 也有一个缺点，即它要求环境中的段落在页面上必须有足够的空白，如果段落文字恰好在一页的末尾，就会在页面上留下大片的空白，这与使用 float 提供的不浮动的图表环境（H 选项）是一样的。此外，picinpar 在图表外面加的间距很小，不方便调整，figwindow 与 tabwindow 环境的语法也与一般的 LaTeX 环境不同。当图形大小或环境中文本内容在页面中位置不合适时，picinpar 偶尔还会造成错误的段落形状，需要在使用中小心调整。

另一个工具是 wrapfig 宏包[14]，它提供了 wrapfigure 和 wraptable 环境，在语法格式上更接近标准 LaTeX 2ε 的 figure 和 table 环境（中括号里面是可选参数）：

① 例子中的 \lipsum 命令需要 lipsum 宏包，它只用来生成一段测试用的文本，没有实际意义。实用中 figwindow 或 tabwindow 环境里是大段的文字。

```
\begin{wrapfigure}[⟨行数⟩]{⟨位置⟩}[⟨外伸长度⟩]{⟨宽度⟩}
    ⟨图内容⟩
\end{wrapfigure}
\begin{wraptable}[⟨行数⟩]{⟨位置⟩}[⟨外伸长度⟩]{⟨宽度⟩}
    ⟨表内容⟩
\end{wraptable}
```

在使用时，⟨图内容⟩及⟨表内容⟩与普通 figure、table 环境的内容相同，可以是任意代码产生的图表，也可以使用 \caption 命令生成标题。在环境后面的段落内容将会沿图表绕排。在这里，⟨位置⟩参数不区分大小写，可以是 l 或 r，即左右两侧，也可以用 o 或 i，表示双面页面的装订内侧和外侧。⟨宽度⟩则指定图表环境所占用的宽度。可选的⟨行数⟩可以指定图表占用的行数，如果留空则会按内容高度自动计算（不过自动计算的结果有时偏大）。⟨外伸长度⟩如果大于 0pt，则图表会向左右侧面伸出版心指定的长度，产生特殊的效果。例如，本段文字开头的表 5.18 就是这样得到的：

表 5.18　向右伸出的绕排表格

甲	乙	丙	丁

```
% \usepackage{wrapfig}
\begin{wraptable}[4]{r}[1.5cm]{4.5cm}
  \centering
  \caption{向右伸出的绕排表格}\label{tag:wraptable}
  \begin{tabular}{|c|c|c|c|}
    \hline 甲 & 乙 & 丙 & 丁 \\ \hline
  \end{tabular}
\end{wraptable}
```

5-3-55

wrapfig 在使用时可以不必指定把多少文字包含进绕排的范围，但它同样也有与 picinpar 类似的问题，即本页中剩下的空间必须足够放下被绕排的图表，否则也将造成难看的分页。在功能上，wrapfig 可以让图伸出版心之外，不过不能把图表放在中间或跳过前几行，因而与 picinpar 有互补的效果。

与 wrapfig 类似的宏包还有 floatflt 宏包[60]，它提供了 floatingfigure 和 floatingtable 的浮动环境，可以把浮动体放在一段开头的左侧或右侧，这里不再详细说明了。floatflt 的独特功能是提供了 \fltitem 命令，试图解决与列表环境共用时产生的问题。

在 TEX 内部，绕排工具都是使用 \parshape 命令的功能（参见 2.2.1 节）配合复杂的盒子操作与计算完成的，这也是为什么在列表环境中无法正常使用绕排功能（因为列表项也是由 \parshape 实现的）。前面介绍的绕排工具都只能产生矩形

的空洞来放置图形，不过 \parshape 提供了更多的可能性：完全可能利用 \parshape 产生更复杂形状的绕排效果。shapepar 宏包的 \cutout 命令就部分地实现了这种复杂绕排的功能，不过具体位置和参数需要手工仔细调整，详细说明请参见手册 Arseneau [15]。这里举一个使用圆形绕排的例子，图形是用 TikZ 画的圆：

5-3-56

```
% \usepackage{shapepar}
% \usepackage{tikz,lipsum}
\cutout{r}(-1cm,1cm)\shapepar[2cm]{\circleshape}%
\begin{tikzpicture}[overlay]
  \filldraw[fill=lightgray] (0.5,-0.5) circle (1);
\end{tikzpicture}\par
\small\lipsum[1]
```

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

一个比较新的 cutwin 宏包[296] 也提供了类似 picinpar、wrapfigure 的绕排功能，同时与 shapepar 一样还支持自定义挖洞的形状。虽然没有 shapepar 一样丰富的预定义形状，不过在处理绕排时其语法相对自然。cutwin 与 shapepar 宏包有命名冲突，宏包的用法这里就不再详细介绍了。

从前面的介绍可以看出，用于图文混排的工具包很多，其特点也各不相同。值得注意的是，picinpar 和 wrapfig 这类绕排的工具都有一些限制，它们通常都不能让显示（displayed）数学公式、各种列表环境正确绕排，也只能应用于大段的普通文本中。同样，被绕排的图表不应该太大，否则特别容易产生不良的分页。因此，这些工具的使用没有 LaTeX 标准浮动体那样自动化，在使用这类工具时，需要在文档编译的最后阶段仔细检查，必要时做出手工调整。

5.4　使用彩色

从概念上说，彩色并非绘图功能，基本的文字、页面部件都可以使用彩色。不过由于原始的 TEX 引擎不支持彩色[①]，有关彩色的功能都是由输出 PS、PDF 格式的 TEX 引擎或驱动（参见 5.2 节）提供的，有关命令也是在绘图相关的扩展宏包中定义，因此我们将彩色放在本章说明。

基本的彩色支持工具是 color 宏包[38]，它是 LATEX 的基本组件，graphics 工具包的一部分。

在 color 宏包中，使用彩色的基本命令是 \color 和 \textcolor：

```
\color{⟨颜色⟩}
\textcolor{⟨颜色⟩}{⟨文字⟩}
```

它们的语法格式与字体选择命令的语法格式非常相似，\color 是声明式命令，它使（同一分组内）后面的内容都使用指定的颜色输出，而 \textcolor 则将参数 ⟨文字⟩ 以指定的颜色输出，例如[②]：

```
% \usepackage{color}
\color{red}红色文字夹杂%
\textcolor{blue}{蓝色}文字
```

红色文字夹杂蓝色文字　5-4-1

除了 \color 和 \textcolor，color 宏包还提供选择页面背景色及彩色盒子的命令，其语法格式如下：

```
\pagecolor{⟨页面颜色⟩}
\colorbox{⟨盒子颜色⟩}{⟨文字⟩}
\fcolorbox{⟨线框颜色⟩}{⟨盒子颜色⟩}{⟨文字⟩}
```

彩色盒子例如：

```
\colorbox{yellow}{黄色盒子} \\
\fcolorbox{black}{green}{黑框绿盒子}
```

黄色盒子


5-4-2

① 在 TEX 发明的年代，彩色印刷远没有今天容易和普及。
② 为示例清晰，使用加粗无衬线体显示彩色文字。

与 \fbox 类似，盒子外框的间距与线框粗细由长度变量 \fboxsep 和 \fboxrule 控制。

彩色命令使用的 ⟨颜色⟩ 参数是预定义的颜色名。在标准的 color 宏包中只有几种原色是预定义的：黑白颜色 black 和 white，色光三原色红 red、绿 green、蓝 blue，以及印刷三原色青 cyan、品红 magenta、黄 yellow。

这三类原色分别使用三种不同的色彩模型（color model[①]）：gray（灰度）、rgb（红绿蓝）和 cmyk（印刷四分色[②]）。在使用颜色时，可以给颜色命令指定模型，然后使用特定色彩模型下的几个分量 [0, 1] 之间的数值来表示具体的颜色，例如：

5-4-3

```
\textcolor[gray]{0.5}{50\% 灰色} \\
\color[rgb]{0.6,0.6,0}暗黄色
```

50% 灰色
暗黄色

此外，还有一种由输出驱动直接支持的 named 名称模型，例如在 Dvips 驱动下，可以给用 usenames 宏包选项直接调用图 5.16 中的各种彩色名，并加以彩色强度，如用

```
% \usepackage[usenames]{color}
% 使用 latex + dvips 编译
\color[named]{Purple,0.6}
```

可以选定 60% 的淡紫色。不过这种方式限于特定的输出驱动，一般很少使用。

用色彩名称选择颜色是比较方便的方式，除了原本的几种原色，可以使用 dvipsnames 宏包选项来获得更多的色彩名，而不必考虑使用的输出驱动，例如：

5-4-4

```
% \usepackage[dvipsnames]{color}
\textcolor{Purple}{紫色文字}
```

紫色文字

dvipsnames 选项调入的色彩名默认以 CMYK 色彩模型给出。

类似的色彩名称也可以由用户自己定义，其语法格式如下：

\definecolor{⟨色彩名⟩}{⟨模型⟩}{⟨分量值⟩}

例如，紫色就可以定义为：

5-4-5

```
\definecolor{Purple}{cmyk}{0.45,0.86,0,0}
```

① 又称色彩空间（color space）。
② CMYK 四个字母分别指青、品红、黄、黑。

用户可以为文档定义自己的整套色彩。

color 宏包只提供了非常基本的彩色支持功能，在许多时候使用不便。xcolor 宏包[120] 在调制复杂的颜色、转换不同色彩模型等方面大大扩展了 color 宏包的功能，同时也保持了 color 宏包原有的命令和语法，成为 color 更为实用强大的代替品。由于使用更为方便，在较新的文档中，经常直接使用 xcolor 代替 color 宏包提供彩色支持。

xcolor 宏包支持更多的色彩模型，使用 rgb、cmy、cmyk、hsb[①]、gray 等模型可以更方便地定义各色色彩，而且将这些色彩模型作为宏包选项，则可以将整个文档的所有色彩都转换到指定的模型去[②]，这对于制作印刷稿特别有用：

```
% 将所有色彩转换为 CMYK 模型
\usepackage[cmyk]{xcolor}
```

5-4-6

xcolor 宏包比 color 宏包支持更多的基本色彩，图 5.15 中的色彩在调用 xcolor 宏包后即可任意使用。同时除了用 dvipsnames 选项可以访问 PostScript 预定义的色彩名称外（见图 5.16），还可以使用 svgnames 和 x11names 访问 SVG 格式或是 UNIX X11 库预定义的大量色彩名称。

尤其有用的是，xcolor 还支持颜色表达式的记法，常用的有：

半色调：　⟨颜色⟩!⟨百分数⟩
混合色：　⟨颜色⟩!⟨百分数⟩!⟨颜色⟩
互补色：　-⟨颜色⟩

这可以方便地表示出 50% 的紫色或是更复杂的将不同颜料按比例混合的中间色调：

```
\textcolor{purple!70}{淡紫色}

{\color{blue!60!black}60\% 蓝与 40\% 黑混合的深蓝色}

\colorbox{-red}{青色与红色互补}
```

5-4-7

淡紫色

60% 蓝与 40% 黑混合的深蓝色

青色与红色互补

*本节以下内容初次阅读可略过。

① HSV 色彩模型指色调（hue）、饱和度（saturation）、亮度（brightness）三个分量的色彩模型。

② 不同模型之间的转换有时是失真的，例如灰度 gray 模型不可能完整地表示红色。

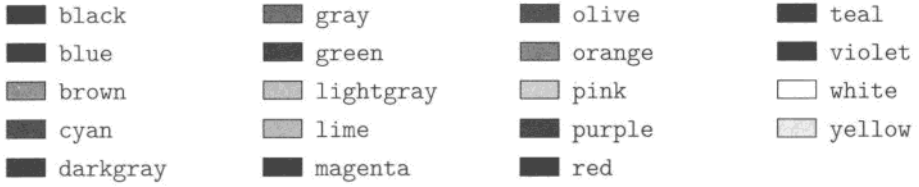


图 5.15 xcolor 基本可用的色彩

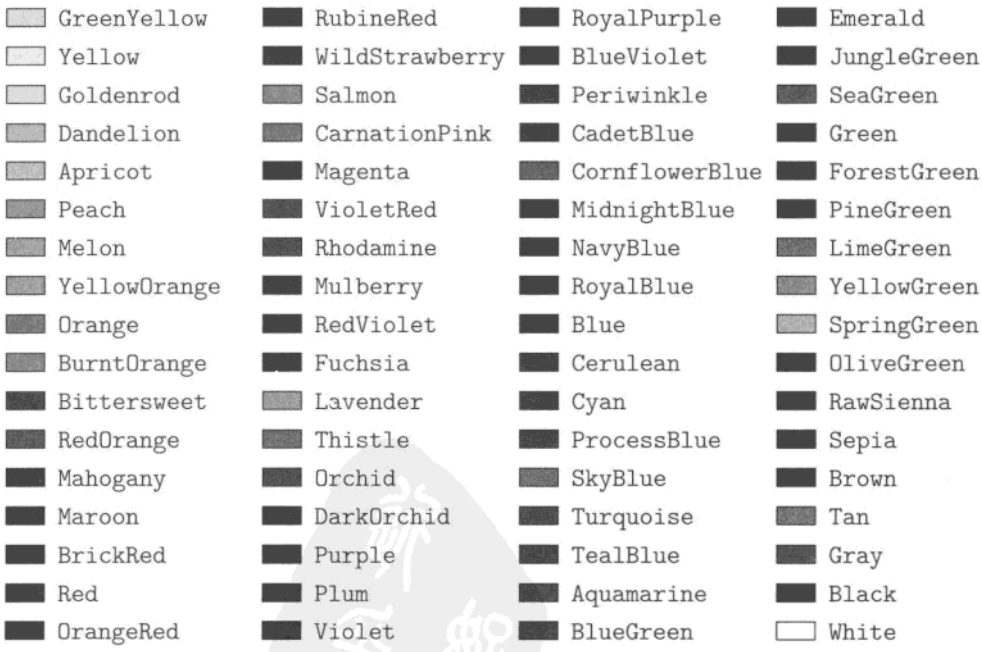


图 5.16 PostScript 预定义的色彩名称，可通过 dvipsnames 选项以任意驱动使用

xcolor 宏包也提供了许多新的命令来支持更丰富的命令，例如 \colorlet 可以使用色彩表达式来定义新色彩名：

```
\colorlet{darkred}{red!50!black}
\textcolor{darkred}{定义暗红色}
```

定义暗红色　5-4-8

更多相关的命令可以参见 xcolor 的宏包手册 Kern [120]。

color 与 xcolor 宏包都不提供透明颜色的支持。使用 pdfTeX 引擎时，可以用 transparent 宏包[180] 实现颜色透明度的支持，它提供了 \transparent 与 \texttransparent 命令，其用法与 \color 和 \textcolor 类似。transparent 宏包不支持 pdfTeX 以外的其他图形驱动，不过 LaTeX 中更为复杂的绘图语言 PSTricks 和 TikZ（参见 5.5.2 节）都支持透明色，如果需要可以调用它们实现。

5.4.1　彩色表格

下面我们回到 LaTeX 的表格，来看看彩色功能是如何应用到表格中的。

彩色表格是由 colortbl 宏包[50] 来实现的。使用 colortbl 可以方便地设置表列、表行、单个单元格或表格线的颜色。

colortbl 基于 color 宏包，调用 colortbl 时会自动调用 color 宏包，如果需要复杂的色彩，则可以需要另外调用 xcolor 宏包。同时 colortbl 也依赖 array 宏包，对表列和表格线的颜色设置需要用到 array 宏包的语法（参见 5.1.6 节）。

设置表列背景颜色的命令是 \columncolor，其语法格式如下：

\columncolor[⟨模型⟩]{⟨色彩⟩}[⟨左侧外伸⟩][⟨右侧外伸⟩]

\columncolor 用在 >{...} 格式说明符中，设置表列的背景色。一般只使用 ⟨色彩⟩ 一个参数，例如：

```
% \usepackage{colortbl}
% \usepackage{xcolor}
\begin{tabular}{>{\columncolor{gray}}c >{\columncolor{lightgray}}c}
  深 & 浅 \\
  darker & lighter \\
\end{tabular}
```

5-4-9

*本节内容初次阅读可略过。

深	浅
darker	lighter

可选的参数 ⟨左侧外伸⟩ 和 ⟨右侧外伸⟩ 是指表列在文字之外还要向两侧伸出的宽度，默认是 \tabcolsep，可以保证表列能被背景色完整填充。对于特殊表列，可能需要修改外伸的长度，例如：

5-4-10

```
\begin{tabular}{|c|@{}>{\columncolor{lightgray}[0pt][\tabcolsep]}c|}
  表列 & 左紧右松 \\
\end{tabular}
```

表列	左紧右松

\rowcolor 命令则用来设置表行颜色，直接把它用在表格一行的开头即可：

5-4-11

```
\begin{tabular}{|ccc|}
\hline \rowcolor{lightgray} A & B & C \\
  一 & 二 & 三 \\\hline % 保持白色
\end{tabular}
```

A	B	C
一	二	三

要改变单个单元格的背景色，则可以使用 \cellcolor 命令，例如：

5-4-12

```
\begin{tabular}{cccc}
  No & No & \cellcolor{lightgray}Yes & No \\
  \cellcolor{lightgray}Yes & No & No & No \\
\end{tabular}
```

No	No	Yes	No
Yes	No	No	No

彩色表格可以利用背景色块来区分不同的行列，一般不使用表格线。表格线的整体颜色可以由 \arrayrulecolor 来设置，它与 \color 命令类似，会影响其后的所有横线与竖线的颜色，同时它的效果是全局的，所以如果后面不再需要这种颜色，则应该恢复原来的颜色。如果用在表格中间，则它只会影响到此命令之后的所有表格线的颜色。\doublerulesepcolor 命令与 \arrayrulecolor 类似，只是用来设置表格双线之间空隙的颜色，默认是黑色，例如：

```
% \usepackage{xcolor, colortbl}
\arrayrulecolor{gray}
\doublerulesepcolor{lightgray}
\begin{tabular}{|c|c|}
\hline\hline 灰色表线 & 浅灰色间隙 \\
\arrayrulecolor{black}\hline
以下为原色 & 表线 \\
\doublerulesepcolor{white}\hline\hline
\end{tabular}
\arrayrulecolor{black} % 恢复默认值
```

灰色表线	浅灰色间隙
以下为原色	表线

5-4-13

不过 \arrayrulecolor 对于 makecell、booktabs、arydshln 等宏包引入的新表线命令不起作用，使用时需要注意。

xcolor 宏包进一步扩展了彩色表格行的设置语法。给 xcolor 宏包加上 table 选项，xcolor 就行自动调用 colortbl 宏包，同时支持如下的命令：

\rowcolors[⟨横线命令⟩]{⟨起始行⟩}{⟨奇数行色彩⟩}{⟨偶数行色彩⟩}
\rowcolors*[⟨横线命令⟩]{⟨起始行⟩}{⟨奇数行色彩⟩}{⟨偶数行色彩⟩}

这些命令用在 tabular 或 array 环境之前，可以让表格从 ⟨起始行⟩ 开始，在奇数行和偶数行使用不同的背景色，同时在每一行前后执行可选的 ⟨横线命令⟩ 画出横线。⟨横线命令⟩ 可以使用 \hline，这样每行表格都会自动画出横线。对于是带星号的版本，则 ⟨横线命令⟩ 则只在交错背景色的行执行。

\rowcolors 命令非常适合格式规律的数据表格，交错的背景色适合可以让读者很容易分清不同的行。在多数情况下，彩色表格也不需要画出横线，例如：

```
% \usepackage[table]{xcolor}
\begin{table}[htbp]
  \centering
  \rowcolors{2}{black!20}{black!10} % 交错的表行
  \begin{tabular}{crrr}
  \rowcolor{black!30} % 第一行的表头单独设置背景色
    项目 & 数值 & 数值 & 数值 \\
    A & 10 & 20 & 30 \\
    B & 20 & 15 & 40 \\
```

```
    C & 15 & 25 & 37
  \end{tabular}
\end{table}
```

5-4-14

项目	数值	数值	数值
A	10	20	30
B	20	15	40
C	15	25	37

xcolor 还提供了 \showrowcolors 和 \hiderowcolors 命令，用来手工打开和关闭表行交错背景色的机制。此外，\rowcolors 的 (横线命令) 与 arydshln 有冲突，在使用 arydshln 宏包时，直接使用 \rowcolors 的 (横线命令) 会重复画线，也可以改用带星号的形式，并在最后用 \hiderowcolors 宏包解决此问题，如：

```
% \usepackage[table]{xcolor} % 将调入 colortbl
% \usepackage{arydshln} % 在 colortbl 后面使用
\rowcolors*[\hline]{1}{black!20}{black!10}
\begin{tabular}{|c:r|}
  A & 10 \\
  B & 20 \\
\hiderowcolors
\end{tabular}
```

5-4-15

A	10
B	20

当使用 colortbl 设置表行背景色时，如果用到了 multirow 宏包的 \multirow 产生跨行文字，在后面生效的彩色行的彩条可能会覆盖前面的文字，例如：

```
\begin{tabular}{|cc|} \hline
\multirow{2}*{Test} & foo \\
\rowcolor{lightgray}& bar \\\hline
\end{tabular}
```

Test foo
bar

解决办法是把 \multirow 放在最后一行输出，此时需要向上跨越负数行：

```
\begin{tabular}{|cc|} \hline
 & foo \\
\rowcolor{lightgray}
\multirow{-2}*{Test} & bar \\\hline
\end{tabular}
```

Test	foo
	bar

5-4-16

练 习

5.11 colortbl 提供了表格背景色和表格线颜色的设置命令，但如何设置表格中文字的颜色呢？

5.5　绘图语言

5.5.1　Xy-pic 与交换图表

Xy-pic 是老牌的数学图形包，距今已有约 20 年的历史，功能丰富。尽管在很多方面已经被其他一些绘图语言所代替，但它目前仍是使用最广泛的 LaTeX 绘图宏包之一，特别是用于绘制数学交换图表。

在文档中载入 Xy-pic 通常使用

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

即以 all 选项载入 xy 包，其中 all 选项表示加载各种常用的功能[①]。

Xy-pic 默认以字符形式输出，也就是说所有的图形都是使用许多特殊字体中的符号拼起来的。这种纯字符的输出方式可以保证对旧输出设备的兼容性，但图形质量欠佳。对于一般输出 PDF 格式的文档，则可以加上 pdf 选项[②]，得到质量更好的输出，即

```
\usepackage[all,pdf]{xy}
```

Xy-pic 中最常用的是基于矩阵的图形，许多文字元素以矩阵的形式排列，并加上一些连线和箭头。数学交换图表、有限状态自动机都属于这种图形。Xy-pic 矩阵命令是 \xymatrix，它带有一个参数，参数中的矩阵语法和 matrix 环境类似，矩阵的内容将以数学模式排版，例如：

① 但并非 Xy-pic 所有的功能。事实上它加载了 curve, frame, tips, line, rotate, color 扩展和 matrix, arrow, graph 特性。

② pdf 选项用于 pdfLaTeX、XeLaTeX 引擎和 dvipdfm(x) 驱动，而如果用 Dvips 驱动则使用 ps 选项。

5-5-1

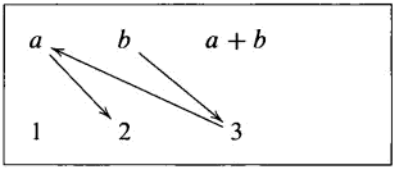
```
\xymatrix{
  a & b & a+b \\
  1 & 2 & 3 \\
}
```

$$\begin{array}{ccc} a & b & a+b \\ 1 & 2 & 3 \end{array}$$

在矩阵中可以使用 \ar 命令表示连线箭头。\ar 命令后面是箭头指向的相对方向，方向可以使用 u、d、l、r（上、下、左、右）及它们的组合。也可以使用双引号表示矩阵中的绝对坐标。使用 ⟨坐标⟩;⟨坐标⟩ 则可以直接给出连线的起点和终点。例如：

5-5-2

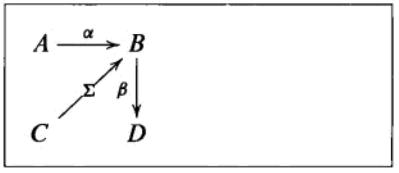
```
\xymatrix{
  a & b\ar[rd] & a+b \\ % 指向右下方
  1 & 2 & 3\ar"1,1"     % 指向 (1,1)
  \ar"1,1";"2,2" % 直接从 (1,1) 到 (2,2)
}
```



在箭头后面可以使用上标或下标，在箭头“上下”方添加标签，也可以用 | 打断连线在中间添加标签。这里的“上下”方向是以向右的箭头为准的，实际为箭头所指方向相对的左侧和右侧，例如：

5-5-3

```
\xymatrix{
  A\ar[r]^{\alpha} & B\ar[d]_{\beta} \\
  C\ar[ur]|{\Sigma} & D \\
}
```



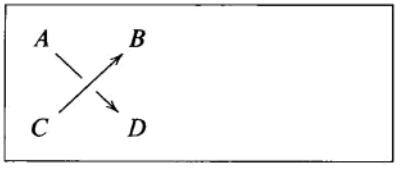
使用 \hole 表示一个空洞，一般用 |\hole 的方式表示有间断的连线，如用

```
$\xymatrix@1{ A\ar[r]|\hole & B }$
```

就可得到 $A \longrightarrow B$。空洞常常用于交叉的连线，例如：

5-5-4

```
\xymatrix{
  A\ar[rd]|\hole & B \\
  C\ar[ru] & D
}
```



可以使用 < 和 > 来修饰连线的标签，把标签放在连线的起点或终点位置，修饰符 << 和 >> 则表示靠近起点和终点的位置，还可以使用 (⟨因子⟩) 直接指定标签在连线上的相对位置，其中 ⟨因子⟩ 在 0 到 1 之间。例如：

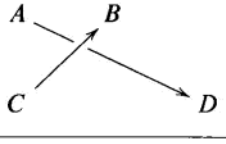
```
\xymatrix{
  A \ar[r]^>{f} & B \\
  C \ar[r]^>>{g} & D \\
  E \ar[r]^(0.6){h} & F
}
```

$$A \xrightarrow{f} B \qquad C \xrightarrow{g} D \qquad E \xrightarrow{h} F$$

5-5-5

Xy-pic 还支持使用 !{⟨甲⟩;⟨乙⟩} 的语法来表示 ⟨甲⟩、⟨乙⟩ 两个方向连线的交点，例如：

```
\xymatrix{
  A \ar[drr]|!{[d];[r]}\hole & B & \\
  C \ar[ur] & & D
}
```



5-5-6

\xymatrix 生成的数学图形可以放在任意环境中。通常把它放在显示公式的环境中，与普通的数学公式一样，例如：

```
\begin{equation}
\begin{gathered} \xymatrix{
  S\ar[r]^{f_s} \ar[d]_{\lambda}
    & T\ar[d]^{\bar\lambda} \\
  S' \ar[r]_{f_{s'}} & T' \\
} \end{gathered}
\end{equation}
```

$$\begin{array}{ccc} S & \xrightarrow{f_s} & T \\ {\scriptstyle\lambda}\downarrow & & \downarrow{\scriptstyle\bar\lambda} \\ S' & \xrightarrow[f_{s'}]{} & T' \end{array} \tag{5.1}$$

5-5-7

这里为了让公式的编号垂直居中，在 \xymatrix 外面套了一层 gathered 环境，否则编号将与矩阵第一行对齐。

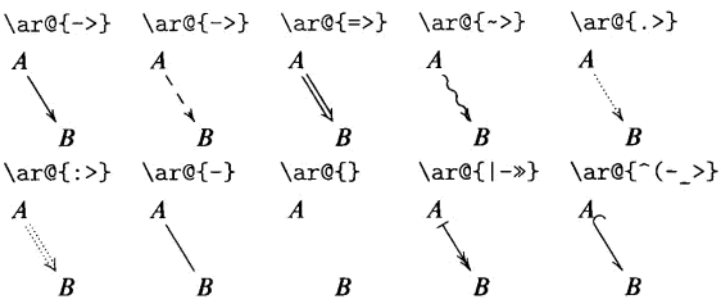
也可以把 Xy-pic 矩阵夹在普通正文中，这时需要把它放在数学模式里，保证正确的基线位置。对于单行矩阵可以在 \xymatrix 后面加上 @1，缩小矩阵元素的距离：

```
映射 $\xymatrix@1{A\ar[r]^{f} & B}$
是同态。
```

映射 $A \xrightarrow{f} B$ 是同态。

5-5-8

在箭头命令 \ar 后面可以加上 @ 开头的样式说明来选择连线和箭头的样式，例如：



样式由中间的连线和箭头的前后端点组成，中间的连线可以是直线、虚线、双线、波浪线、点线等样式。箭头也可以是尖角、圆角、平角等样式，或是通过上下标调整位置的样式。连线和箭头都可以为空，不同的样式可以组合成多种复杂的形式。

除了直接连线，Xy-pic 也可以画出弯曲的箭头连线，其语法是在 \ar 命令后面加上 @/⟨曲线说明⟩/，基本的用法就是 @/^/ 和 @/_/，即沿箭头向左或向右弯曲，例如：

5-5-9

```
\[ \xymatrix{
  A \ar@/^/[r]^{\phi} & B \ar@/^/[l]^{\psi}
} \]
```

$$A \overset{\phi}{\underset{\psi}{\rightleftarrows}} B$$

更复杂的曲线样式则可以用 @(⟨出⟩,⟨入⟩) 指定曲线发出和射入的方向，例如画有向图的自环：

5-5-10

```
\[ \xymatrix{
  A \ar[r] & B \ar@(ur,dr)
} \]
```

$$A \longrightarrow B \circlearrowright$$

在箭头命令 \ar 后面加上 @<⟨偏移量⟩> 可以画出向一侧平移的箭头，这特别适用于在两个对象之间画双箭头，例如：

5-5-11

```
\[ \xymatrix{
  A \ar@<.5ex>[r]^f &
  \ar@<.5ex>[l]^g B
} \]
```

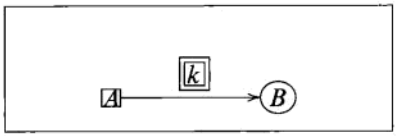
$$A \underset{g}{\overset{f}{\rightleftarrows}} B$$

在 Xy-pic 中，不仅连线可以有多种样式，矩阵的元素和标签也可以有不同的样式。使用 * 可以引入一个带有修饰符的对象，其语法格式如下：

*⟨修饰⟩{⟨文本⟩}

例如：

```
\[ \xymatrix@=2cm{
  *[F]{A} \ar[r]^*+[F=]{k} & *+[o][F]{B}
} \]
```



5-5-12

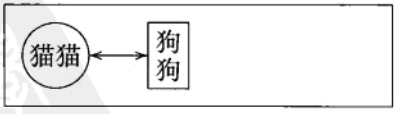
其中 [F] 表示加框，[F=] 加双框，[o] 表示使用圆形框，+ 表示增加对象与边框的间距；Xy-pic 中常见的修饰符见表 5.19。这里在 \xymatrix 后面用 @=⟨距离⟩ 来设置相邻矩阵元素的间距。

表 5.19　Xy-pic 常见的对象修饰符

语法	说明
+	增加边距
+<⟨长度⟩>	增加指定 ⟨长度⟩ 的边距
+=	扩展边距到正方形，使水平与垂直边距相等
-	减少边距
-<⟨长度⟩>	减少指定 ⟨长度⟩ 的边距
-=	减少边距到正方形，使水平与垂直边距相等
!	不居中
[o]	设置边框为椭圆形
[l] [r] [u] [d]	左、右、上、下对齐
[F] [F=]	单线框　双线框
[F.] [F--]	点线框　虚线框
[F-,] [F-<3pt>]	阴影框　圆角矩形框

之前 Xy-pic 中的标签都是数学公式，使用 \txt{...} 命令则可以得到文本模式的 Xy-pic 对象。与简单地使用盒子 \mbox 或是 amsmath 的 \text 不同，使用 \txt 还可以在内容中用 \\ 换行，使用 \txt<⟨宽度⟩> 还可以让文字按指定宽度自动换行，例如：

```
\xymatrix{
  *+++=[o][F]\txt{猫猫} \ar@{<->}[r] &
  *+[F]\txt{狗\\狗}
}
```

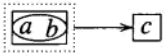


5-5-13

这里组合使用了 +、+=、[o] 和 [F] 修饰符来画出比文字稍大的圆形边框。

命令 \composite{... * ...} 可以用来把两个不同的 Xy-pic 对象重叠在一起组合成一个对象，例如：

5-5-14
```
\xymatrix{
  *+[F.]{\composite{*+[o][F]{a\quad} * *+[F]{\quad b}}} \ar[r]
    & *+[F]{c}
}
```

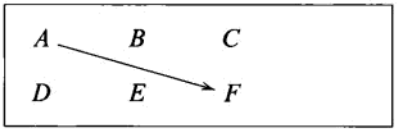


在 \xymatrix 命令后面也可以使用如下修饰符来调整矩阵元素的间距：

- @=⟨长度⟩ 设置元素间距
- @R=⟨长度⟩ 设置行间距
- @C=⟨长度⟩ 设置列间距
- @M=⟨长度⟩ 设置元素的默认边距
- @W=⟨长度⟩ 设置元素的默认宽度
- @H=⟨长度⟩ 设置元素的默认高度
- @L=⟨长度⟩ 设置标签的边距

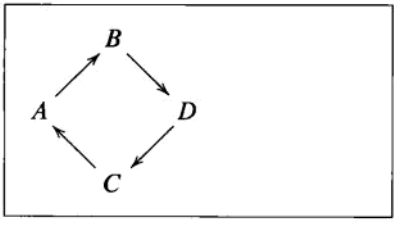
其中的等号 = 还可以替换为 +、+=、- 和 -=，其意义与表 5.19 一致，例如：

5-5-15
```
\xymatrix@R=2ex{
  A \ar[drr]& B & C \\
  D & E & F
}
```



在 \xymatrix 后面直接加上 u, d, ul, ur 等方向，则表示将矩阵的连线按此方向旋转，使原来向右的方向指向给定的坐标方向，例如：

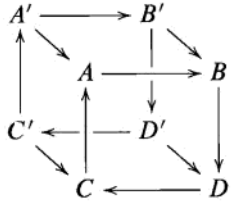
5-5-16
```
\xymatrix@ru{
  A \ar[r] & B \ar[d] \\
  C \ar[u] & D \ar[l]
}
```



有关 Xy-pic 的更多用法说明可参见宏包的官方指南 Rose [217] 与参考手册 Rose and Moore [218]。

练 习

5.12 试绘制下面的立方形交换图：



5.5.2 PSTricks 与 TikZ 简介

这一节我们介绍两个功能强大的绘图语言包。一个是老牌的 PSTricks，它利用 PostScript 语言强大的计算和图形功能，可以在 TEX 中画出非常复杂的图形；另一个则是较为晚近的 TikZ 宏包，它具有与 PSTricks 相近的绘图能力，同时具有更强的可移植性和更人性化的语言界面。

绘图语言 PSTricks 与 TikZ 绘制的都是使用数学坐标描述的矢量图形。在多数情况下，文章中的插图用专门的绘图软件做好就可以了，并不需要精确的坐标和复杂的命令描述。不过在一些特定场合，如几何图形、函数图像、树形结构等，这种命令式作图的方式仍然具有不可替代的作用（见例 5-3-48 中的图灵机就是用 TikZ 画的）。而 TEX 中的绘图宏包可以方便地使用 TEX 语句进行公式标注，因而也在这种场合非常常用。

这两个绘图宏包的功能十分复杂，详细介绍其中任何一个都将超出本书的全部篇幅。在这里我们只能举两个简单的例子，分别使用两个不同的宏包绘制图 5.17，以期给读者一个较为直观的印象。管中窥豹，或可略见一斑。当然，如果要实际使用它们绘图，这里的介绍是远远不够的，仍然需要阅读专门的手册或专著。

*本节内容初次阅读可略过。

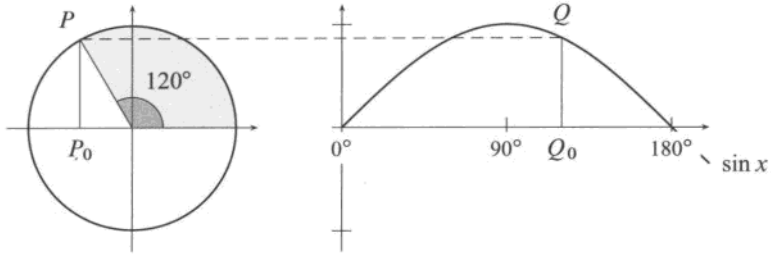


图 5.17 正弦函数与单位圆

5.5.2.1 PSTricks

PSTricks 由一个核心 pstricks[299]、一个附加包 pstricks-add[215] 和许多以 pst- 开头的功能模块包组成，其中附加包也会载入几个常用的功能模块。在绘制最简单的几何图形时，只需要加载核心的 pstricks 包就可以了：

```
\usepackage{pstricks}
```

本节的例子则还使用了数据与函数图功能包 pst-plot[303] 以及结点连接功能包 pst-node[302] 的一些功能。这两个功能包都包含在 pstricks-add 之中，因此本节的例子只需要载入

5-5-17
```
\usepackage{pst-plot,pst-node}
```

或者

5-5-18
```
\usepackage{pstricks-add} % 同时载入内核与常用的功能模块
```

就可以编译了。直接载入所有需要的模块，或是载入 pstricks-add，都是典型的加载 PSTricks 的做法。

因为 PSTricks 要大量使用 PostScript 语言，所以通常需要使用 Dvips 作为图形驱动，才能得到最完整的图形效果。在使用 XeTeX 引擎时，PSTricks 会自动调用 GhostScript 转化成需要的格式，也可以直接编译，只是速度稍慢。如果使用 pdfLaTeX 或 LuaLaTeX 编译，则需要再给 pstricks 加上 pdf 选项①，同时在编译时加上 -shell-escape（TeX Live）或 --enable-write18（MiKTeX）选项②，让编译程序自动重新编译生成单独的 .pdf 图片并插入文档。而对于使用 DVIPDFMx 等其他编译方式，则无法这样直接编译，只能在一个单独的图片文档中，用 standalone 文档类[225] 等工具宏包生成单独的图片，然后再插入到主文档中。编译 PSTricks 图形的不同方式见表 5.20。

① 此选项只能直接加给内核包 pstricks，这会调用 auto-pst-pdf 宏包[211]。也可以直接调用 auto-pst-pdf 完成同样的效果。
② 新版本的 MiKTeX 也支持 -shell-escape 选项。

表 5.20　编译 PSTricks 图形的方式（其中 Dvips 驱动是基本的直接支持）

编译方式	图形驱动	需要选项或宏包
latex + dvips + ps2pdf	Dvips	无
xelatex	XeTeX (xdvipdfmx)	无
pdflatex -shell-escape	pdfTeX	pdf 选项
lualatex -shell-escape	LuaTeX	pdf 选项
latex + dvipdfmx	DVIPDFMx	standalone 并单独插图

PSTricks 宏包的大部分绘图命令名都以 ps 开头，例如 \psline 命令就是把一列坐标点用直线连接，坐标可以省略单位，默认单位是 1 cm：

```
直线
\psline(0,0)(1,1em)(1.5,0)
直线
```

直线 直线

5-5-19

不过正如例 5-5-19 所示，直接使用绘图命令产生的图形不占任何位置，通常要把所有绘图命令放在 pspicture 环境中，而在环境开始指明图形所占盒子的左下角、右上角坐标，左下角坐标如果是原点则可以省略：

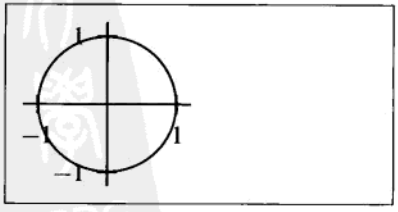
```
直线
\begin{pspicture}(1.5,1em)
\psline(0,0)(1,1em)(1.5,0)
\end{pspicture}
直线
```

直线 直线

5-5-20

现在，我们首先来画图 5.17 左侧的单位圆。坐标轴使用 pst-plot 的\psaxes 绘制，可以分别指定坐标轴的原点坐标 (x_0, y_0) 与左下、右上边界坐标 (x_1, y_1), (x_2, y_2)，其中前两个坐标取原点时都可以省略，而单位圆用 \pscircle 指定圆心和半径画出：

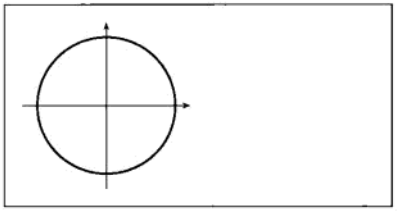
```
\begin{pspicture}(-1.2,-1.2)(1.2,1.2)
  \psaxes(0,0)(-1.2,-1.2)(1.2,1.2)
  \pscircle(0,0){1}
\end{pspicture}
```



5-5-21

不过，上面画出的图形在格式上并不符合我们的要求。在画单位圆时，我们不需要坐标轴上的刻度与数字标注，但需要给坐标轴加上箭头，这就要给 \psaxes 设置 labels 与 ticks 的格式选项与箭头说明；另外，我们还需要让单位圆的线比坐标轴粗一些，用 linewidth 选项设置线宽。在 PSTricks 中，选项可以加在命令后面，也可以使用 \psset 进行统一设置。这里，我们用 \psset 将默认的线条设置为细线，而在画圆时切换成粗线：

5-5-22
```
\psset{linewidth=0.4pt}
\begin{pspicture}(-1.2,-1.2)(1.2,1.2)
  \psaxes[labels=none,ticks=none]
    {->}(0,0)(-1.2,-1.2)(1.2,1.2)
  \pscircle[linewidth=0.8pt](0,0){1}
\end{pspicture}
```



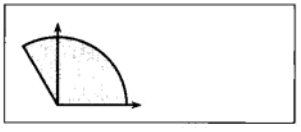
注意，PSTricks 命令后用方括号括起的选项后面可以有空格，但花括号中的箭头说明之后或是坐标之间则不能有空格或换行，必须紧密相连或在行末用注释符 % 断行。

下面来画指定幅角的扇形，这由 \pswedge 命令得到，需要指定原点、半径和起止角度（辐角以度为单位）：

5-5-23
```
\begin{pspicture}(-0.5,0)(1,1)
  \pswedge(0,0){1}{0}{120}
\end{pspicture}
```

实心的形状可以在命令后面加星号 * 得到，例如 \pswedge*(0,0){2em}{0}{30} 就得到◢。不过如果需要更复杂的彩色填充，就需要设置 fillstyle, fillcolor 等选项对填充格式、填充色彩进行控制了，例如：

5-5-24
```
\begin{pspicture}(-0.5,0)(1.2,1.2)
\psaxes[labels=none,ticks=none]{->}(1.2,1.2)
\pswedge[fillstyle=solid,fillcolor=gray,
  opacity=0.2](0,0){1}{0}{120}
\end{pspicture}
```



这里除了设置使用灰色实心填充外，opacity 选项还将扇形填充为半透明的，把不透明度设置为 0.2 可以让先画的坐标轴也显露出来。用同样的方法还可以画出另一个扇形用于标注角度。

下面来看文字的标注。文字标注可以使用一般的 \rput 或更专门的 \uput 命令，把文字按要求放在指定的坐标位置。其中 \uput 命令除了指定坐标位置与要放置的内容，还要指定一个参考方向（角度或是 u, d, l, r 等方向），内容会朝参考方向偏移一小段位置，这特别适合文字的标注。PSTricks 中的极坐标点用 ((半径);(辐角)) 的方式给出。因此，标注图 5.17 中的点 P 的命令就是：

```
\uput[120](1;120){$P$}
```

5-5-25

不过，由于点 P 的坐标会反复使用，利用 pst-node 的功能给点 P 起一个名字，即建立一个坐标结点，会更为方便。建立坐标结点的命令是 \pnode，只需要指定结点的坐标和名称。对于建立好的结点，可以用结点名来代替它的坐标。于是我们有：

```
\pnode(1;120){P}
\uput[120](P){$P$}
```

5-5-26

其效果与例 5-5-25 相同。

点 P_0 则是点 P 在 x 轴上的投影。在 PSTricks 中，并不需要单独计算投影点的坐标，只要使用 (A|B) 的语法，就可以得到点 A 的横坐标与点 B 的纵坐标结合的坐标，因此，点 P_0 就可以定义为：

```
\pnode(1;120){P}
\pnode(P|0,0){P0}
```

5-5-27

pst-node 的 \ncline 命令可以用来用直线连接两个结点。当然，对于坐标结点，\ncline 的效果与 \psline 直接画直线的效果是一样的：

```
\pnode(1;120){P}
\pnode(P|0,0){P0}
\ncline{-}{P}{P0} % 等价于 \psline(P)(P0)
```

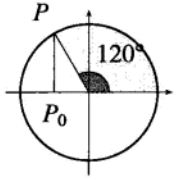
5-5-28

于是，我们把所有这些命令组合在一起，就可以得到下面的代码和图形：

```
\psset{linewidth=0.4pt}
\begin{pspicture}(-1.2,-1.2)(1.2,1.2)
  \psaxes[labels=none,ticks=none]{->}(0,0)(-1.2,-1.2)(1.2,1.2)
  \pscircle[linewidth=0.8pt](0,0){1}
  \pswedge[fillstyle=solid,fillcolor=gray,opacity=0.2]
    (0,0){1}{0}{120}
  \pswedge[fillstyle=solid,fillcolor=gray,opacity=0.5]
```

```
    (0,0){0.3}{0}{120}
  \uput[60](0.3;60){$120^\circ$} % 在扇形中间标注角度
  \pnode(1;120){P}
  \pnode(P|0,0){P0}
  \ncline{-}{P}{P0}              % 正弦线
  \uput[120](P){$P$}
  \uput[d](P0){$P_0$}
\end{pspicture}
```

5-5-29

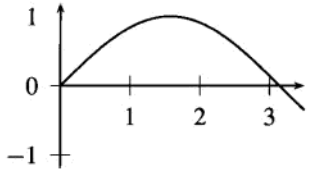


下面我们来看图 5.17 右侧的正弦函数图像，首先不考虑左侧的单位圆，单独画出图的右半边，然后再进行组合。

函数图像可以由 pst-plot 提供的 \psplot 命令产生，它分别需要输入变量 x 的范围与要画的函数。例如要画出正弦函数 $\sin x$ 在 $[0, 3.5]$ 的图像，就可以用：

```
\begin{pspicture}(-1,-1.2)(3.5,1.2)
  \psaxes{->}(0,0)(0,-1.2)(3.5,1.2)
  \psplot{0}{3.5}{x 180 Pi div mul sin} % 三角函数单位是度
\end{pspicture}
```

5-5-30



在这里，\psplot 的函数使用的是 PostScript 语言的后缀表达式，因此正弦函数 $\sin x$ 就应该写成 `x sin`，但因为正弦函数的单位是度而不是这里要求的弧度，所以还需要进行单位转换，$x° = (x \cdot 180/\pi)\,\text{rad}$，也就是使用 `x 180 Pi div mul sin` 来表示 $\sin(x \cdot 180/\pi)$。

这种后缀表达式语法可参见 PostScript 语言的手册[4] 或教程[2、3]。对一般人来说，这种被称为逆波兰记法（Reverse Polish notation）的后缀表达式并不方便使用，因此，