

Chapter 2 Typesetting Text

Du Ang

du2ang233@gmail.com

2017 年 5 月 31 日

目录

1 文章和语言的结构	3
2 断行和断页	3
2.1 合理分段	3
2.2 连字符 (Hyphenation)	4
3 预定义好的字符串	4
4 特殊符号	4
4.1 引号 (Quotation Marks)	4
4.2 短划线 (Dashes) 和连字符 (Hyphens)	4
4.3 波浪线 (Tilde)	5
4.4 斜杠 (Slash)	5
4.5 度 (Degree Symbol)	5
4.6 欧元符号	5
4.7 省略号 (Ellipsis)	5
4.8 连字 (Ligatures)	5
4.9 重音 (Accents) 符号和特殊符号	6
5 国际语言支持/中文排版支持	6
5.1 C _T E _X 的安装	7
5.2 使用 C _T E _X 文档类	7
6 单词之间的空格	8
7 标题、章、节	8
8 交叉引用 (Cross References)	9
9 脚注 (Footnotes)	9
10 强调 (Emphasis)	9

11 环境 (Environments)	10
11.1 Itemize, Enumerate 和 Description	10
11.2 Flushleft, Flushright 和 Center	11
11.3 Quote, Quotation 和 Verse	11
11.4 Abstract	12
11.5 Verbatim	13
11.6 Tabular	14
12 浮动体 (Floating Bodies)	17
13 保护脆弱命令 (Protecting Fragile Commads)	18

1 文章和语言的结构

写文章最重要的一点是要把想法、信息、知识传达给读者，而好的文章结构能够帮助读者更好地查看、感受和理解我们想传达的东西。

L^AT_EX 中最重要的文本单位是段 (paragraph)。一段文字应该只包含一个思想或一种想法。写文章时什么时候分段？应该怎么分段呢？如果要写一个新的想法了，那就另起一段，对应源码中空一行)；否则，可以用换行符 (line breaking) 来继续写原来的想法，对应源码中使用 `\\` 或 `\newline`。

很多人都低估了合理分段的重要性。在 L^AT_EX 中，很多人甚至都不知道什么是分段，有时自己已经新起了一段都不知道。按照我的理解，一般情况下是不需要使用换行符的，在该分段的时候在代码里空一行另起一段就行了。但是在使用公式 (equation) 的时候需要考虑好该使用什么，这时候很容易犯上述的错误。下面是说明该换行还是该另起一段的三个正确示例：

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

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

Kirchhoff' s voltage law can be derived \ldots

% Example 3
\ldots which has several advantages.
\begin{equation}
  I_D = I_F - I_R
\end{equation}
is the core of a very different transistor model. \ldots
```

2 断行和断页

2.1 合理分段

- `\\` 或 `\newline`：断行、不另起一段。`\\` 也在表格、公式等地方用于分行，而 `\newline` 只用于文本段落中。
- `*`：断行、不另起一段、不断页。

- `\newpage` 或 `\clearpage`: 断页。二者有些细微的区别: 一是在双栏排版中 `\newpage` 只起到另起一栏的作用; 二是涉及到浮动体的排版上行为不同。
- `\linebreak[n]`、`\nolinebreak[n]`、`\pagebreak[n]`、`\nepagebreak[n]`: 向 L^AT_EX 建议哪些地方适合断行、断页, 哪些地方不适合断行、断页。n 是数字, 代表适合/不适合的程度, 取值范围为 0 到 4, 默认为 4。数字越大代表程度越高。

一般 L^AT_EX 都会找到最适合断行的地方。但是有些时候——比如它不知道该如何用连字符分割单词的时候, 它可能会让一行文字从段落右边伸出一部分, 然后报出 `overfull hbox` 的警告。这时使用 `\sloppy` 命令可以使单词之间的间距增大来避免这个问题, 但是这时会报出 `underfull hbox` 的警告。大多数情况下这样排版出来的都不太好看。可以使用 `\fussy` 命令恢复 L^AT_EX 的默认方式。

2.2 连字符 (Hyphenation)

对于绝大部分单词, L^AT_EX 都能够找到合适的断词位置, 在断开的行尾加上连字符 -。如果一些单词没能自动断词, 我们可以在单词内手动使用 `\-` 命令指定断词的位置。另外, 也可以使用 `\hyphenation{word list}` 命令来指定使用连字符的位置, 例如 `\hyphenation{FORTRAN Hy-phen-a-tion}`, 其中的 word list 是不区分大小写的。

`\mbox{text}` 或 `\fbox{text}`: 会避免 text 被连字符分开。`\fbox` 比 `\mbox` 多了个可见的框。

3 预定义好的字符串

- `\today`: 2017 年 5 月 31 日 (打印当天日期)
- `\TeX`: T_EX
- `\LaTeX`: L^AT_EX
- `\LaTeXe`: L^AT_EX 2_ε

4 特殊符号

4.1 引号 (Quotation Marks)

- 双引号: ``...text...'`
- 单引号: ``...text...'`

4.2 短划线 (Dashes) 和连字符 (Hyphens)

在 L^AT_EX 中有下面四种横杠:

- `-`: -, 连字符 (hyphen), 用于连接词语
- `--`: –, 短破折号 (en-dash), 常用于连接数字表示起止范围
- `---`: —, 长破折号 (em-dash), 常用于表示意思的转换
- `$-$`: −, 减号 (minus sign)

4.3 波浪线 (Tilde)

- `\~{}`: ~
- `\sim`: ~

4.4 斜杠 (Slash)

- `read/write`: read/write (不允许用连字符拆分)
- `read\slash write`: read/write (允许用连字符拆分)

4.5 度 (Degree Symbol)

- `$30\,\sim{\circ}\mathrm{C}$1`: 30 °C
- `30 \textcelsius`: 30 °C
- `86 \textdegree`F: 86 °F

4.6 欧元符号

要想使用欧元符号, 需要先在导言区通过 `\usepackage{textcomp}` 命令导入宏包, 然后再使用 `\texteuro` 命令输出欧元符号。如果所用的字体不包含欧元符号或者想用别的字体的欧元符号, 可以通过 `\usepackage[official]{eurosym}` 导入 `eurosym` 宏包, 然后用 `\euro` 输出官方的欧元符号。用 `gen` 来替换 `official` 参数可以使用和当前字体匹配的欧元符号。

- `\texteuro`: €
- `\euro`: €

4.7 省略号 (Ellipsis)

L^AT_EX 提供了命令 `\ldots` 来生成省略号, 相对于直接输入三个点的方式更为合理。`\ldots` 和 `\dots` 是两个等效的命令。

- Apples, bananas, ...: Apples, bananas, ...
- Apples, bananas, `\ldots`: Apples, bananas, ...
- Apples, bananas, `\dots`: Apples, bananas, ...

4.8 连字 (Ligatures)

有些相邻的字母在排版时会连接起来, 可以通过 `\mbox{}` 命令避免它们相连。

- `ffshfilfluffia`: ffshfilfluffia (相连的情况)
- `f\mbox{f}shf\mbox{i}lf\mbox{l}uf\mbox{f}f\mbox{i}a`: ffshfilfluffia (没有相连的情况)

¹这里的 `\`, 会输出空格

4.9 重音 (Accents) 符号和特殊符号

示例代码：

```
\emph{\=a} \emph{\'a} \emph{\v a} \emph{\` a}
```

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

示例输出：

\bar{a} \acute{a} \check{a} \grave{a}

Hôtel, naïve, élève,
smørrebrød, ¡Señorita!,
Schönbrunner Schloß Straße

重音符号和特殊符号命令列表：

```
\`o    \'o    \^o    \~o
\=o    \.o    \"o    \c c
\u o    \v o    \H o    \c o
\d o    \b o    \t oo
\oe    \OE    \ae    \AE
\aa    \AA
\o    \O    \l    \L
\i    \j    !`    ?`
```

重音符号和特殊符号输出结果列表：

\grave{o} \acute{o} \hat{o} \tilde{o}

\bar{o} \dot{o} \ddot{o} \S

\check{o} \breve{o} \acute{o} \mathfrak{Q}

\mathfrak{O} \mathfrak{O} \mathfrak{Oo}

\mathfrak{oe} \mathfrak{E} \mathfrak{ae} \mathfrak{AE}

\mathfrak{a} \mathfrak{A}

\emptyset \varnothing \mathfrak{l} \mathfrak{L}

\mathfrak{l} \mathfrak{J} \mathfrak{i} \mathfrak{j}

5 国际语言支持/中文排版支持

L^AT_EX 对其他很多语言提供了支持。babel 宏包可以用于对各种语言进行适配。其他语言暂时也用不到，这里就记录一下如何让 L^AT_EX 支持中文。

使用 L^AT_EX 排版中文有两种方式,一种是使用 x_eCJK 宏包,另一种是使用 C_T_EX 宏包和文档类,推荐使用后者。C_T_EX 宏包和文档类是对 CJK 和 x_eCJK 等宏包的进一步封装。文档类包括 ctexart、ctexrep、ctexbook,分别是对 L^AT_EX 的三个标准文档类 article、report、book 的封装,对 L^AT_EX 的排版样式做了许多调整,以切合中文排版风格。最新版本的 C_T_EX 宏包/文档类甚至支持自动配置字体。

5.1 C_T_EX 的安装

C_T_EX 宏集依赖的宏包和宏集已被最常见的 T_EX 发行版 T_EXLive 和 MiK_T_EX 所收录。如果本地安装的 T_EXLive 或 MiK_T_EX 不是完全版本,就需要通过这两个发行版提供的宏包管理器来安装宏包。

T_EXLive 的宏包管理器是 tlmgr。在 Linux 系统上,一般需要 sudo 权限才能正确地执行 tlmgr 的功能。

直接使用 sudo tlmgr [arg] 时,可能会提示找不到 tlmgr 或没有这个命令。乍一看,情况比较尴尬:不加 sudo 没有权限,加了 sudo 反而找不到命令了。经过上网搜索,在一个帖子²里找到了解决方法。原来,sudo 有一种内置的保护机制,只会使用安全的环境变量 PATH。如果 T_EXLive 的路径不在 sudo 的安全环境变量内,它就找不到相关的命令。可以在终端执行 sudo gedit /etc/sudoers,然后将 T_EXLive 的路径添加到 sudo 的 secure_path 中。在我的 Ubuntu 16.04 上,添加后结果如下(后面的原有路径省略,不同路径用 : 隔开):

```
Defaults secure_path="/usr/local/texlive/2016/bin/x86_64-linux:/usr/local/sbin:..."
```

不能用 sudo 执行 tlmgr 的问题解决后,在终端中依次执行以下命令,以更新 tlmgr 宏包管理器、已安装的所有宏包、安装 C_T_EX 宏集。

```
sudo tlmgr update --self
sudo tlmgr update --all
sudo tlmgr install ctex
```

5.2 使用 C_T_EX 文档类

C_T_EX 宏集提供了四个中文文档类:ctexart、ctexrep、ctexbook 和 ctexbeamer,分别对应 L^AT_EX 的标准文档类 article、report、book 和 beamer。使用它们的时候,需要将涉及到的所有源文件使用 UTF-8 编码保存。

下面是使用 ctexart 文档类编写的一个例子:

```
\documentclass[UTF8]{ctexart}
\begin{document}
```

中文文档类测试。你需要将所有源文件保存为 UTF-8 编码。

你可以使用 XeLaTeX、LuaLaTeX 或 upLaTeX 编译,也可以使用 (pdf)LaTeX 编译。

推荐使用 XeLaTeX 或 LuaLaTeX 编译。

```
\end{document}
```

C_T_EX 预定义的字库中的中文字体已经基本够用,包括宋体 (\songti)、黑体 (\heiti)、楷书 (\kaishu)、仿宋 (\fangsong) 等。更多 C_T_EX 的使用参考《C_T_EX 宏集手册》³。

² sudo does not find tlmgr, <https://tex.stackexchange.com/questions/203874/sudo-does-not-find-tlmgr>

³ 《C_T_EX 宏集手册》, <http://mirror.unl.edu/ctan/language/chinese/ctex/ctex.pdf>

6 单词之间的空格

为了使输出更美观、更具可读性， \LaTeX 可能会在不同单词之间或句子末尾插入更多空格。 \LaTeX 默认句子以句点 (periods)、问号 (question marks) 或者感叹号 (exclamation marks) 结尾。但是如果句点跟在一个大写字母后面，它不会认为这是句子结尾，因为大写字母后面跟句点往往是缩略词。

用户可以通过具体的命令来改变上面的默认设定。一个斜杠跟一个空格会产生一个不会被扩大的空格；一个波浪线 (~) 会产生一个既不能被扩大、也不能从这里断行的空格；在句点前使用 `\@` 命令，不管这个句点是不是跟在大写字母后面，都会指定这个句子到句点就结束。使用 `\frenchspacing` 命令可以强制不在一个句子后面插入多余的空格。如果使用 `\frenchspacing` 命令就没必要再用 `\@` 了。

示例代码：

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

示例输出：

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

7 标题、章、节

文档类 `article` 中有以下几种分层次结构的命令：

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

`\part{...}` 命令也可以把文档分为多个部分，但它不会影响 `section` 和 `chapter` 的编号。

和 `article` 文档类相比，在 `report` 和 `book` 中，可以使用 `\chapter{...}`。

由于 `article` 文档类中不包含 `chapter`，所以可以很方便地把 `article` 作为 `chapter` 插入 `book` 文档类。章节空隙、编号等由 \LaTeX 自动完成。

下面是两个比较特殊的情况：

- `\part` 命令不会影响 `chapter` 或 `section` 的编号
- `\appendix` 命令没有任何参数，会把 `chapter`（对于 `report`、`book`）或 `section`（对于 `article`）的数字编号转换成字母编号。

`\tableofcontents` 命令可以用于建立目录，目录就会在这条命令所在的位置生成。一般新写的文档需要编译两次才能正确生成目录，必要的时候 \LaTeX 也会提示需要编译三次。

上面提到的分章节的命令都有一个加星号的版本，就是在原来的命令名称后面加一个星号，成为稍有不同的新命令。例如 `\section{Help}` 命令，加星号之后的命令为 `\section*{Help}`。加星版本的章节命令对应的标题不会显示在目录里，也不会被编号。

有时候章节的标题太长，这会导致其在目录里显示不佳。可以通过下面的命令在真正的标题前选择添加一个参数，指定在目录中显示的标题。

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

整个文档的标题是通过 `\maketitle` 命令产生的。在调用 `\maketitle` 命令之前，文档标题的内容需要由 `\title{...}`、`\author{...}`、`\date{...}`（可选）等参数指定。在 `\author{...}` 命令的参数中，可以用 `\and` 来间隔多个作者名字。

所有标准文档类都提供了一个 `\appendix` 命令将正文和附录分开，使用 `\appendix` 后，最高一级章节改为使用拉丁字母编号，从 A 开始。

L^AT_EX 2_ε 在 book 文档类有以下三个额外的命令，可以进行前言、正文、后记的结构划分。这三个命令还可和 `\appendix` 命令结合，生成有前言、正文、附录、后记四部分的文档。

- `\frontmatter` 前言部分，放置在文档主体的最开始 (`\begin{document}`)，它会把页码变成罗马数字，其后的 `\chapter` 不编号
- `\mainmatter` 正文部分，页码为阿拉伯数字格式，从 1 开始计数，其后的章节编号正常
- `\backmatter` 后记部分，页码格式不变，继续正常计数；其后的 `\chapter` 不编号

8 交叉引用 (Cross References)

在写作时经常要用到对图片、表格等的交叉引用，在 L^AT_EX 中可以用下面的命令完成：

```
\label{marker}, \ref{marker}, \pageref{marker}
```

其中，marker 是由用户自行定义的标识符。

示例代码：

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

示例输出：

A reference to this subsection looks like: “see section 8 on page 9.”

9 脚注 (Footnotes)

可以使用 `\footnote{...}` 命令来添加脚注，脚注应该紧跟在它注解的词或句子（包括标点符号）后面。

由于脚注会分散读者的注意力，所以尽量在文章主体说清楚，少用脚注。

10 强调 (Emphasis)

以前用打印机打字的时候，习惯把重要的词用下划线来强调，这在 L^AT_EX 中可以通过 `\underline{...}` 命令来实现。但在印刷书籍中，一般通过 `\emph{...}` 命令，使用意大利字体 (*italic font*) 进行强调。

`\emph{...}` 命令使用意大利字体进行强调并不是绝对的，这还要结合具体的语境。

示例代码：

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

示例输出：

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

11 环境 (Environments)

环境的典型命令为 `\begin{\emph{environment}} text \end{\emph{environment}}`，其中 *environment* 是环境的名字。

环境可以相互嵌套，例如：

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

11.1 Itemize, Enumerate 和 Description

示例代码：

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

示例输出：

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

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

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

11.2 Flushleft, Flushright 和 Center

`flushleft`、`flushright` 和 `center` 环境分别会使段落左对齐、右对齐和居中。

示例代码：

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

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

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

示例输出：

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

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

At the centre
of the earth

11.3 Quote, Quotation 和 Verse

`quote` 环境适合引用一些名言、重要的词句、示例等。

示例代码：

A typographical rule of thumb for the line length is:

```
\begin{quote}
```

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

```
\end{quote}
```

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

示例输出：

A typographical rule of thumb for the line length is:

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

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

`quotation` 环境和 `verse` 环境很像。但由于 `quotation` 环境对每一段都会缩进，所以适合引用比较长的、一般有几段的内容；而 `verse` 环境很适合引用诗歌，利用 `\\` 或空行来分行。

示例代码：

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

```
\begin{flushleft}
```

```
\begin{verse}
```

Humpty Dumpty sat on a wall:\\

Humpty Dumpty had a great fall.\\

All the King' s horses and all

the King' s men\\

Couldn' t put Humpty together

again.

```
\end{verse}
```

```
\end{flushleft}
```

示例输出：

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

Humpty Dumpty sat on a wall:

Humpty Dumpty had a great fall.

All the King' s horses and all the King' s men

Couldn' t put Humpty together again.

11.4 Abstract

在科学刊物中，一般会以一段摘要开头，让读者对这篇文章有一个整体的认知，这是一个惯例。 \LaTeX 的 `abstract` 环境就是用于写摘要的，一般用于 `article` 文档类。

示例代码：

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

11.5 Verbatim

在 `verbatim` 和 `verbatim` 之间的所有文本都会原封不动地打印出来，文本中的 `LaTeX` 命令不会被执行，所以可以用 `verbatim` 环境来插入一段代码。

如果要插入行间代码，可以用 `\verb+text+` 命令。其中的 `+` 只是分隔符，可以由用户随意指定，但不能用字母、`*` 或空格，习惯上用 `|`。

示例代码：

```
The \verb|\ldots| command \ldots

\begin{verbatim}
  10 PRINT "HELLO WORLD ";
  20 GOTO 10
\end{verbatim}
```

示例输出：

The `\ldots` command ...

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

`verbatim` 环境和 `verb` 命令都各自有一个加 `*` 的版本，区别是会把空格显示成 `_`。

示例代码：

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

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

示例输出：

like_ _ _this_ :->_

the_starred_version_of_the_verbatim_environment_emphasizes_the_spaces
in_the_text.

`verbatim` 环境和 `\verb` 命令对符号的处理比较复杂, 一般不能用在其它命令的参数里, 否则多半会出错。

`verbatim` 宏包优化了 `verbatim` 环境的内部命令, 并提供了 `\verbatiminput` 命令用来直接读入文件生成代码环境。`fancyvrb` 宏包提供了可定制格式的 `verbatim` 环境; `listings` 宏包更进一步, 可生成关键字高亮的代码环境, 支持各种程序设计语言的语法和关键字; 本篇笔记使用的是 `minted` 宏包, 代码可以有彩色高亮。详情请参考各自的帮助手册。

11.6 Tabular

`tabular` 环境可以用于排版表格, \LaTeX 会自动地调整表格每一列的宽度。具体命令如下:

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

其中, `table spec` 参数决定了表格的格式, `l`、`r`、`c` 分别会使单元格内容左对齐、右对齐、居中, 不折行; `|` 会绘制竖线; `p{width}` 会使单元格固定宽度为 `width`, 可以自动折行; 可以通过 `@{...}` 来在单元格前后插入任意的文本, 但同时它会消除单元格前后额外添加的间距。

表格中每行的单元格数目不能多于列格式里 `l/c/r/p` 的总数 (可以少于这个总数), 否则出错。

`pos` 参数用于指定表格相对于环绕为基线的垂直位置, 可以为 `t`、`b` 和 `c`, 分别代表顶部 (top)、底部 (bottom) 和中间 (center)。

在 `tabular` 环境内, `&` 会跳到下一列, `\\` 会新起一行, `\hline` 会插入一条水平线。使用 `\cline{i-j}` 会插入一部分水平线, 其中 `i` 和 `j` 是列的编号。

示例代码 1:

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

示例输出 1:

7C0	hexadecimal
3700	octal
11111000000	binary
1984	decimal

示例代码 2:

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

```
\hline
\end{tabular}
```

示例输出 2:

Welcome to Boxy's para- graph. We Sincerely hope you'll all enjoy the show.

如果想让表格的某一列按小数点对齐, 又不使用额外的宏包 (`dcolum`), 可以采用一种比较折中的方法: 把该列的所有小数分为整数部分和小数部分, 各作为一列, 然后在 `tabular` 环境中使用 `@{.}` 作为整数列和小数列的分隔符。整数部分和小数部分之间不要忘了用 `&` 隔开, 因为它们现在是两列了。为了在这两列的表头仅显示一个标签, 可以使用 `\multicolumn` 命令。

示例代码 3:

```
\begin{tabular}{c r @{.} l}
Pi expression      &
\multicolumn{2}{c}{Value} \\
\hline
$\pi$              & 3&1416 & \\
$\pi^{\pi}$        & 36&46 & \\
$\pi^{\pi^{\pi}}$   & 80662&7 & \\
\end{tabular}
\quad
\begin{tabular}{|c|c|}
\hline
\multicolumn{2}{|c|}{Ene} \\
\hline
Mene & Muh! \\
\hline
\end{tabular}
```

示例输出 3:

Pi expression	Value	Ene	
π	3.1416	Mene	Muh!
π^{π}	36.46		
$(\pi^{\pi})^{\pi}$	80662.7		

有时需要为整列修饰格式, 比如整列改变为粗体, 如果每个单元格都加上 `\bfseries` 命令会比较麻烦。`array` 宏包提供了辅助格式 `>` 和 `<`, 用于给列格式前后加上修饰命令。辅助格式甚至支持插入 `\centering` 等命令改变 `p{width}` 列格式的对齐方式, 一般还要加额外的命令 `\arraybackslash` 以免出错。因为 `\centering` 等对齐命令会破坏表格环境里 `\\` 换行命令的定义, `\arraybackslash` 用来恢复之。如果不加 `\arraybackslash` 命令, 也可以用 `\tabularnewline` 命令代替原来的 `\\` 实现表格换行。

示例代码 4:

```
\begin{tabular}>{\itshape}r<{*}l}
\hline
italic & normal \\
column & column \\
\hline
\end{tabular}

\begin{tabular}
>{\centering\arraybackslash}p{9em}}
\hline
Some center-aligned long text. \\
\hline
\end{tabular}
```

示例输出 4:

<i>italic</i> *	normal
<i>column</i> *	column
Some center-aligned long text.	

有时 L^AT_EX 默认的表格会比较拥挤, 可以通过改变 `\arraystretch` 和 `\tabcolsep` 参数来调节。

示例代码 5:

```
\begin{tabular}{|l|}
\hline
These lines\\
are tight\\
\hline
\end{tabular}

{\renewcommand{\arraystretch}{1.5}
\renewcommand{\tabcolsep}{0.2cm}
\begin{tabular}{|l|}
\hline
less cramped\\
table layout\\
\hline
\end{tabular}}
```

示例输出 5:

These lines
are tight

less cramped
table layout

如果只想增加表格中某一行的高度, 可以通过添加一条不可见的“支柱”(bar) 来实现。即使用 `\rule` 命令, 并把宽度设置为 0。

示例代码 6:

```
\begin{tabular}{|c|}
\hline
\rule{1pt}{4ex}Pitprop \ldots\ \ % 宽度不为 0, 可见
\hline
\rule{0pt}{4ex}Strut\ \ \ \ \ \ % 宽度为 0, 不可见
\hline
\end{tabular}
```

示例输出 6:

Pitprop ...
Strut

如果想排版较长的表格, 可以使用 `longtable` 环境。在 `booktabs` 宏包中, 还有很多其他的命令可以美化表格。

12 浮动体 (Floating Bodies)

一般的刊物、出版品都会有各种图表, 由于这类元素不能跨页, 所以需要特殊对待。有一种方法是每当遇到一个图或表在当前页放不开的时候, 就新起一页。这种方法会让页面有比较多的留白, 看起来会不美观。

一种比较好的解决方法是当前页放不开这个图表时, 把它们“浮动”到后面的页面去, 当前页面用正文进行填充。 \LaTeX 提供了两种浮动体环境: `figure` 和 `table`, 分别针对图片和表格。任何放在 `figure` 和 `table` 环境中的东西都会被视为是浮动体。

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

所有的浮动体环境都会有一个叫 `placement specifier` 的可选的参数, 可以用于指定这个浮动体被允许移动到哪里。下面是 `placement specifier` 参数的可选值 (默认为 `tbp`, 优先级按照 `h-t-b-p` 排列, 与顺序无关):

- `h`, 当前位置 (代码所处的上下文)
- `t`, 顶部, 如果是当页排版可能出现在代码之前
- `b`, 底部
- `p`, 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, 有文本的页面称为文本页 (text page)

参数的时候会失效，甚至引起报错。需要在脆弱命令前加上 `\protect` 命令来保护它们，这样它们就可以作为移动参数了。

`\protect` 命令只会影响紧跟在它之后的一个命令，一般多加了 `\protect` 命令也不会出什么问题。