

## 第四章 显示文本

有许多方法可用来显示或强调文本：改变字体样式或字体尺寸，居中，缩进，标志段落，等等。L<sup>A</sup>T<sub>E</sub>X为我们提供了这些最常见显示形式的命令。

### §4.1 改变字体

在排版印刷中，一组特定大小和样式的字母、数字及符号的组合称为一种字体。在 L<sup>A</sup>T<sub>E</sub>X 中用于文本主体的标准字体是一种直立的罗马字体，其由文档开头处的 `\documentclass` 或 `\documentstyle` 声明来定义其平均权重。可用的基本尺寸是 10, 11 或 12pt，分别相应于尺寸选项 `10pt`（默认值），`11pt` 或 `12pt`。（注意一英寸大约等于 72.27 点，而一厘米大约等于 28.45 点。）小括号 `()` 就占据了等于字体尺寸的高度和深度。

三种标准尺寸的视觉效果相应于这三个数字的比率而言是相当大的：

This is an example of the 10 pt font. `()`

And this is the 11pt font for comparison. `()`

And finally this is a sample of 12 pt font. `()`

#### §4.1.1 强调

在用打字机打出来的手稿中，强调文本的最简单方法就是用 下划线。在印刷时，制版工人通常要把用下划线的文本转化为斜体。在 L<sup>A</sup>T<sub>E</sub>X 中要从标准文本切换为强调形式，只需要用声明 `\em` 或命令 `\emph`<sup>2ε</sup>。

声明 `\em` 的作用同下面将要讲解的其它字体声明一样：改变当前字体，直到被其它相应的声明取消（它可以是 `\em` 自身），或者当前环境的结束（2.2 节）。也可以只用一对大括号 `{...}` 来创建一个环境。这是一种最简单的强调一小段文本的方法，请见示例：

This is the easiest way to `{\em emphasize}` short ...

而在 L<sup>A</sup>T<sub>E</sub>X 2<sub>ε</sub> 中提供了命令 `\emph`，这是一种强调一两个单词的更合逻辑的方法：

A more logical method of `\emph{emphasizing}` a word ...

注意它们的差别，声明的作用是当局部环境用右大括号结束时而终止，而命令只对其包含在大括号内的参数有作用。另外一种更精细的差别就是命令 `\emph` 自动插入倾斜校正（见 3.5.1 节），而在声明 `\em` 中则必须人工加入这一校正。

声明和命令都切换进入一种强调字体。这也就是说如果当前字体是直立的，那就切换为斜体，而如果当前字体是 *slanted*，则就切换为一种直立字体。

可以嵌套强调，而且其非常易懂：

The `{\em first}`, second, and `{\em third font switch}`

The `{\em first, {\em second, and {\em third font switch}}}`  
的结果都是 ‘The *first*, *second*, and *third font switch*’。

### §4.1.2 字体尺寸的选择

在 L<sup>A</sup>T<sub>E</sub>X 中可以用下面的声明来改变字体尺寸：

<code>\tiny</code>	<small>smallest</small>	<code>\Large</code>	larger
<code>\scriptsize</code>	<small>very small</small>	<code>\LARGE</code>	even larger
<code>\footnotesize</code>	<small>smaller</small>	<code>\huge</code>	still larger
<code>\small</code>	<small>small</small>	<code>\Huge</code>	largest
<code>\normalsize</code>	<small>normal</small>		
<code>\large</code>	<small>large</small>		

上面所有的声明都是相对于在文档类选项中的标准尺寸而言的。在本书中，标准尺寸为 10pt，这可以用 `\normalsize` 来选择该尺寸。

字体尺寸声明与所有其它声明的行为是一样的：它立即发挥作用，直到被另一个相反的声明所取消，或者当前环境结束。如果在大括号 `{...}` 内调用，声明的作用就只局限于大括号内，这就如同一个无名环境：

```
normal {\large large \Large larger} normal again
normal large larger normal again
```

在 L<sup>A</sup>T<sub>E</sub>X 2.09 中字体尺寸声明也同时把所有其它字体属性（4.1.3 节）重设为默认值，也就是直立罗马平均权重。与之相反，在 L<sup>A</sup>T<sub>E</sub>X 2<sub>ε</sub> 中，这些属性是不变的。试做一比较：

```
2.09 \sl slanted {\Large larger}  slanted larger
2ε  \sl slanted {\Large larger}  slanted larger
```

如果在 `\documentclass` 的地方用的是 `\documentstyle`，那么这些尺寸声明的作用同 L<sup>A</sup>T<sub>E</sub>X 2.09 中的一样，这就是为了与原来文档兼容才如此设计的。

用上面的命令来改变字体尺寸，也同时自动改变了行间距。对于每一种字体尺寸，都有一个对应的自然行间距 `\baselineskip`。可以在任何时候改变其值。例如，如果自然行距是 12pt，命令 `\setlength{\baselineskip}{15pt}` 就会把其增为 15pt。

在一段结束时有作用的 `\baselineskip` 命令被用来包装整段。这就意味着如果在一段中对 `\baselineskip` 进行了几次修改，那么只会考虑最后那次修改。

每次修改字体尺寸，`\baselineskip` 都会被重设为相应的自然行距。由 `\setlength` 所做的设置也就无效了。

为了创建一个对所有字体尺寸都适用的行间距修改，那就必须用因子 `\baselinestretch`，其正常值为 1。实际上真正的行间距是

```
\baselinestretch×\baselineskip
```

这样对所有的字体尺寸就会保持相应的间距。用户可以随时改变其值：

```
\renewcommand{\baselinestretch}{因子}
```

这里的 `因子` 是一个浮点数。取值 1.5 就意味着把行间距（基线与基线之间的距离）从相应于所有尺寸的自然间距增加了 50%。

然而，`\baselinestretch` 的新值只有当又进行了一次字体尺寸改变时才会发挥作用。为了在当前字体尺寸中实现一个新值，那就必须切换到一个新的字体尺寸，然后马上切换回来。如果当前字体尺寸为 `\normalsize`，那么序列

```
\small\normalsize
```

就可以达到所需的效果。可以用任何命令来取代 `\small`。

并不是在所有的尺寸中都可以用所有的字体样式。如果选择了一种不可能的字体尺寸和样式组合，那么  $\text{\LaTeX}$  就会发出一个警告信息，并在打印时告诉你用何种字体取代了该字样。

### §4.1.3 字体属性 2 $\epsilon$

字体的尺寸只是用来描述这种字体的几种属性中的一个。做为  $\text{\LaTeX}$  2 $\epsilon$  内部集成的一部分，在新字体选择框架 (NFSS) 中，可以与 8.5 节所描述的那样，只用这些属性来选择字体。然而，对于普通用户，就需要用声明和相应的命令来简化这一过程。

对于由  $\text{\TeX}$  和  $\text{\LaTeX}$  提供的计算机现代字体，具有下列的属性和及其对应值：

**Family**（族）指一般的综合样式。传统印刷中的族有诸如 *baskerville*, *Bodoni*, *Times Roman*, *Helvetica* 等等的名称。标准  $\text{\LaTeX}$  2 $\epsilon$  的安装中有如下的声明提供了三种族：

```
\rmfamily 切换（回）到一种罗马字体；
```

```
\ttfamily 切换到一个打字机 (typewriter) 字体；
```

```
\sffamily 切换到一个 sans serif 字体。
```

**Shape**（形状）指的是字体的构成。在标准安装中可以使用的形状声明为

```
\upshape 切换（回）到一种直立字体；
```

```
\itshape 切换到一种斜体；
```

```
\slshape 选择一种叫 slanted 的字体；
```

```
\scshape 切换到小形大写字母 (SMALL CAPS)。
```

**Series**（序列）指的是字体的宽度和 / 或权重（黑度）。可用的声体为

```
\mdseries 切换（回）到平均权重；
```

```
\bfseries 切换到黑体字样字体。
```

上述并没有穷尽所有的字体属性，但其涵盖了最标准的字体，尤其是计算机现代字体。对于其它字体，特别是 PostScript 字体，还存在其它的属性。详情

请见 8.5 节。

这些声明的用法同其它的一样，通常包含在一对大括号 `{...}` 内，如 `{\scshape Romeo and Juliet}`，结果为 **ROMEO AND JULIET**。对于很长一段文本，可以用相应的环境：

```
\begin{ 字体样式 }... 新字体中的文本 ...\end{ 字体样式 }
```

这样的开始结束切换是非常明了的。其中的 字体样式，可以上面的字体命令，只要去掉 `\` 就可以了。

改变字体的一种属性，而其它的保持不变，这样就可以得到许多种组合。

（然而，这并非意味着对每种可能的组合，都存在着一种字体；如果不存在的话，那就会用另一种来替代。）如果我们首先用 `\bfseries` 选择一种黑体序列，接着用 `\slshape` 改为 *slanted* 形状，那么我们就得到一种黑色的 *slanted* 字体：

```
normal and {\bfseries bold and
{\slshape slanted} and back} again.
```

的结果为： normal and **bold and *slanted* and back** again.

最后要指出一点，声明 `\normalfont` 要把所有的属性（除了尺寸）重设成其默认值，即罗马字体，直立和平均权重。有时为了使字体有效，调用这条命令是非常有用的。

#### §4.1.4 字体命令 2ε

对于上面所列的每一种字体声明，都有相应的字体命令，它们给其参数中的文本取指定属性的字体。

Family:	<code>\textrm{ 文本 }</code>	<code>\texttt{ 文本 }</code>	<code>\textsf{ 文本 }</code>
Shape:	<code>\textup{ 文本 }</code>	<code>\textit{ 文本 }</code>	<code>\textsl{ 文本 }</code>
	<code>\textsc{ 文本 }</code>		
Series:	<code>\textmd{ 文本 }</code>	<code>\textbf{ 文本 }</code>	
默认值:	<code>\textnormal{ 文本 }</code>		
强调:	<code>\emph{ 文本 }</code>		

注意这里也包含了 `\emph` 命令，其相应的声明是 `\em`。`\textnormal` 的参数被设置成 `\normalfont` 所选择的标准字体。

这些改变字体的命令可以有来作用于一小段文本或单个单词，这样就比在一个环境中放一个声明要合理得多。前面那个例子现在变为：

```
normal and \textbf{bold and \textsl{slanted}
and back} again.
```

结果仍然为： normal and **bold and *slanted* and back** again.

同 `\emph` 命令一样，当在直立和斜体 /*slanted* 字体之间切换时，这些字体命令会自动地加进必要的倾斜校正（3.5.1 节）。当会出现这种校正时，可

以加入命令 `\nocorr` 来取消它，如

```
\textit{some italics\nocorr} without correction
{\slshape italics \nocorr\textup{without} correction}
```

请记住：字体属性声明和命令只适用于  $\text{\LaTeX} 2_{\epsilon}$ ，而不适用于 2.09 版本。

#### §4.1.5 旧字体声体 2.09

为了与  $\text{\LaTeX} 2.09$  兼容，仍旧保留了两字母的字体声明（它们很早就是  $\text{\TeX}$  的一部分）。

```
\rm Roman      \it Italic      \sc SMALL CAPS
\bf Bold face   \sl Slanted     \sf Sans Serif
\tt Typewriter
```

这些命令与新的相应命令有不同的作用方式：它们是严格地选择一种新字体，而不是只改变某一属性，但是不会改变当前尺寸。这就好像同时调用了 `\normalfont` 声明一样。

`{\bf text}` 等于 `{\normalfont\bfseries text}`

注意：在  $\text{\LaTeX} 2_{\epsilon}$  之前的两个 NFSS 版本中，这些两字母声明被定义成新的长声明。为了处理用那些系统编写的文档，需要在导言中加进软件包 `newfont`（8.8.3 节）。

#### §4.1.6 其它字体

很可能你所在的计算中心或者所用的 PC  $\text{\TeX}$  软件包具有不只上面所列的那些字体形状和尺寸。可以查查局部指南或者软件包指令手册。如果确实如此，那么在  $\text{\LaTeX}$  文档中可以使用它们，方法是要么直接引用它们的名称，要么如果它是建立在 NFSS 之上的，可以使用它们的属性。

要用名称上载一种字体，可以用命令

```
\newfont{\fnt}{名称 scaled 因子} 或
\newfont{\fnt 字体}{名称 at 尺寸}
```

它把字体赋给新的字体命令名 `\fnt`。在前一种情形中，因子是一个数，其 1000 倍于放缩因子，这个因子用来对字体的基本或设计尺寸进行放大或缩小。在第二种情形中，字体被放缩到指定的尺寸。要安装一种 *slanted, sans serif* 字体，尺寸为 20.74pt，名称为 `\sss`，那么我们可以用命令

```
\newfont{\sss}{cmssi17 at 20.74pt}
```

上载 `cmssi17 at 20.74pt`。现在声明 `\sss` 就同 `\rm` 和 `\it` 改变字体一样把字体切换为上述字体，只是基线分隔没有改变。

与之相应的，也可以用属性来进行新字体声明（见 8.5.4 节）：

```
\DeclareFixedFont{\sss}{OT1}{cmss}{m}{sl}{20.74}
```

实际上，如果不知道当前所想用的字体编码和 `\sffamily`，或者不想那么精

确地声明尺寸，那么也可以如下给出：

```
\DeclareFixedFont{\sss}{\encodingdefault}{\sfdefault}
    {m}{s1}{20}
```

（默认值在 8.5.3 节解释。）

### §4.1.7 字符集与符号

T<sub>E</sub>X 和 L<sup>A</sup>T<sub>E</sub>X 显示所用的字符集是由程序实现的。对于不同的版本，约有 400 到 800 个的字体文件，分别相应于不同的字样、尺寸和放缩比例，每个文件由 128（将来会是 256）个字符或符号组成。

每个字符集分别存贮在它们自己的文件中。在附录 E 中列出了 75 个标准字符文件的名称，其中很多就是打印出来的。

在字符集中的每个符号都是通过一个介于 0 到 127（或者 255）之间的数来编址的。命令

```
\symbol{ 数 }
```

会生成当前字体中内部标识号等于 数 的符号。在当前字体中符号  $\iota$  的内部编号为 62，因此可以用命令 `\symbol{62}` 打印出它来。标识号也可以用八进制（前缀 '）或十六进制（前缀 "）数给出。因此符号命令 `\symbol{28}`，`\symbol{'34}` 与 `\symbol{"1C}` 是一样的，都会生成  $\phi$ 。

`\symbol` 命令也可以用来生成还没有为它定义其它命令的符号，例如，`{\tt\symbol{'40}\symbol{'42}\symbol{'134}}` 得到 `"\`。

E.6 节给出了不同符号族中标识号与字符的对应关系。

## §4.2 居中与缩进

### §4.2.1 居中文本

环境

```
\begin{center} 第一行 \\ 第二行 \\ ... 第 n 行 \end{center}
```

把由 `\\` 命令分开的各节文本居中排列。（可以用 `\\[ 长度]` 来插入可以省略的额外行间距。）如果文本的长度超过一行，那就会用一致的单词间距把它分成几行，除了最后一行外，尽可能地使它充满每一行。不会有断词现象出现。

在一个环境内部，可以用命令 `\centering` 来居中后接文本，同样用 `\\` 做为行分隔符。声明的作用到该环境结束时为止。

单独一行文本可以把它做为 T<sub>E</sub>X 命令 `\centerline{文本}` 的参数来居中排列。

### §4.2.2 单边调整

环境

```
\begin{flushleft} 第一行 \\ 第二行 \\ ... 第 n 行 \end{flushleft}
```

```
\begin{flushright} 第一行 \\ 第二行 \\ ... 第 n 行 \end{flushright}
```

使得文本向左（`flushleft`）或向右（`flushright`）对齐。如果这一段文本在一行中放不下，那就把它断成相同单词间距的几行，这同 `center` 环境一样。而且也不会出现断词。

也可以在环境内部用如下的声明达到同样的效果：

```
\raggedleft 代替 flushleft 环境，而
```

```
\raggedright 代替 flushright 环境。
```

### §4.2.3 两边缩进

可以用下面的环境来使一段文本的两边都缩进一定的长度：

```
\begin{quote} 文本 \end{quote}
```

```
\begin{quotation} 文本 \end{quotation}
```

为了与通常的文本区分开，在这段文本的上方和下方插入了额外的间距。

要显示的文本可具有任意的长度；它可以是一句话，也可以是一个段落，甚至于几个段落。

段落与通常一样用空行表示分段，但是在显示文本的开始和结束并不需要空行，因为无论如何这些地方总要插入竖直间距。

上面两种引用环境的区别在于：

在 `quotation` 环境中，段落是用第一行具有缩进来标识的，而在 `quote` 环境中则是用更多的竖直间距来表示的。

当前文本就是用 `quotation` 环境生成的，而上面的文本则是用 `quote` 环境生成的。

只有当通常的文本段落是有第一行进行缩进来标识时 `quotation` 环境才具有意义。

### §4.2.4 诗歌缩进

为了排版诗歌、韵文，需要两边都进行缩进，这就环境

```
\begin{verse} 诗 \end{verse}
```

就比较合适了。例如：

节与节之间用空行分开

而每一节中用 `\\` 分开不同的行。

如果一行太长，以致于超过行宽，那就会对它进行左右调整，并在

下一行上继续，而且进一步向里缩进。

上述的缩进框架可以嵌套使用。在一个 `quote` 环境中可以有其它的 `quote`，`quotation` 或 `verse` 环境。每嵌套一次，文本的两边都会加上额外的缩进，

而且其上下都有竖直间距；然而随着嵌套层数的增加，这些量是减小的。最多允许嵌套六层。

**练习 4.1:** 在练习文件中使用 `quote` 和 `quotation` 环境，并插入一些文本，即把一些文本包围在如下结构中：

```
\begin{quote}. . . . . \end{quote} 或
\begin{quotation}. . . . . \end{quotation}
```

**练习 4.2:** 生成一个叫 `poem.tex` 的新文件，在 `verse` 环境中输入一首你钟爱的诗。选择 12pt 做为标准的字体尺寸，斜体字样。在 `verse` 环境之前用稍大的黑体字样（如 `\Large\bfseries`）写出诗的题目。把诗的题目右对齐。

注意：记住可以在一个环境中包含改变字体样式或尺寸的声明，其作用到环境结束时为止。

**练习 4.3:** 再生成另一个叫 `title.tex` 的文件。还记得可以用 `titlepage` 环境以生成一个单独的标题页吧？如果不记得了，就回头看看 3.3.1 节。用该环境创建一个标题页，自己选择字体尺寸和样式，所有项都要居中。

注意：在 `titlepage` 环境中，你也能用 `center` 环境，但是只用 `\centering` 声明也就足够了，因为当 `titlepage` 环境结束时，其作用也就终止了。

用命令 `\\[长度]` 选择单独一行的间距，这里要指定一个适当的长度。记住文本第一行之前的竖直间距必须用 `*` 形式命令 `\vspace*[长度]` 才能得到（见 3.5.3 节）。

尝试在标题页上不同部分（如题目、作者姓名，地址）用不同的字体尺寸和样式，一直到结果令你满意为止。

把你自己设计的标题页与练习 3.8 中的结果做一对比。如果你更喜欢你自己的作品，不妨用它取代标准练习文件 `titlepage` 环境中的命令 `\title`，`\author` 和 `\maketitle`。

### §4.3 列表

要生成有格式的列表，有三种可以使用的环境：

```
\begin{itemize}      列表文本 \end{itemize}
\begin{enumerate}    列表文本 \end{enumerate}
\begin{description}  列表文本 \end{description}
```

在这些环境中每一个的列表文本都要相对于左边界进行缩进，并且具有一个标签或记号。标签的类型就与所用的环境是什么有关了。生成标签的命令是 `\item`。

#### §4.3.1 itemize 样例

- 每一项前面有一个黑点，即所谓的 *bullet*，做为标签。



- 每一项中的文本可以具有任意长度。标签位于文本的第一行开头处。
- 相邻项之间具有额外的垂直间距分开。

上述结果是用如下输入生成的：

```
\begin{itemize}
\item 每一项前面有一个黑点，即所谓的 \emph{bullet}，做为标签。
\item 每一项中的文本可以具有任意长度。标签位于文本的第一行开头处。
\item 相邻项之间具有额外的垂直间距分开。
\end{itemize}
```

#### §4.3.2 `enumerate` 样例

1. 标签由相邻的数字组成。
2. 每调用一次 `enumerate` 环境，标签都是从 1 开始编号。

上面的例子是由如下输入的文件生成的：

```
\begin{enumerate}
\item 标签由相邻的数字组成。
\item 每调用一次 \texttt{enumerate} 环境，标签都是从 1 开始编号。
\end{enumerate}
```

#### §4.3.3 `description` 样例

**目的** 这个环境适用于定义一组单词或表达式。

**例子** 关键词做为标签，每一项由分类或解释组成。

**其它用途** 也可以用于参考文献中的作者列表。

上面的样例是如下生成的：

```
\begin{description}
\item[目的] 这个环境适用于定义一组单词或表达式。
\item[例子] 关键词做为标签，每一项由分类或解释组成。
\item[其它用途] 也可以用于参考文献中的作者列表。
\end{description}
\item[选项] 命令中包含一个可省略的参数，它以黑体形式成为标签。
```

#### §4.3.4 嵌套列表

上面给出的列表环境也可以彼此嵌套，深度可达四层。所用标签要视嵌套层数而定。缩进总是相对于包围它的列表左边界来确定的。下面就是一个四重的 `itemize` 环境：

- 第一层的标签是一个黑点，即 `bullet`。
- 第二层的标签就是一条长破折号。
- \* 第三层的是一个星号。
- 而第四层的标签就是单个点了。

· 同时，随着嵌套层数的增加，竖直间距在减少。

\* 返回第三层。

– 返回第二层。

• 现在我们又回到 `itemize` 环境的第一层了。

`enumerate` 环境是类似的，随着嵌套层数的增加，标签的样式也在改变：

1. 第一层的标签是阿拉伯数字后跟一个句号。
  - (a) 在第二层，它就是包在小括号 () 内的小写字母。
    - i. 第三层就用小写罗马数字后跟句号来标识。
      - A. 而在第四层，就用大写字母。
        - B. 可以用后面一节的方法来改变标签样式。
    - ii. 返回到第三层了。
  - (b) 返回到第二层了。
2. 现在又回到 `enumerate` 环境的第一层了。

下面是一个混合类型的嵌套例子：

- 第一层的 `itemize` 标签是 bullet 。
  1. 这里的标签是阿拉伯数字，因为这是 `enumerate` 环境的第一层。
    - 这是嵌套的第三层，但却是第二层的 `itemize` 。
      - (a) 这是全部嵌套的第四层，但只是第二层 `enumerate` 环境。
      - (b) 因此标签是包在小括号 () 内的小写字母。
    - 在这一层的标签是一长破折号。
  2. 每个列表都至少应该有两项。

• 在 `\item` 命令前的空行是没有用的。

上面这个混合列表是用如下输入得到的：

```
\begin{itemize}
  \item 第一层的 \texttt{itemize} 标签是 bullet 。
  \begin{enumerate}
    \item 这里的标签是阿拉伯数字，因为这是...
    \begin{itemize}
      \item 这是嵌套的第三层，但却是...
      \begin{enumerate}
        \item 这是全部嵌套的第四层，但只是...
        \item 因此标签是包在小括号 () 内的小写字母。
      \end{enumerate}
      \item 在这一层的标签是一长破折号。
    \end{itemize}
    \item 每个列表都至少应该有两项。
  \end{enumerate}
\end{itemize}
```

```
\item 在 \item 命令前的...
\end{itemize}
```

练习 4.4: 利用 `itemize` 和 `enumerate` 环境生成同上面例子那样的嵌套列表, 只是命令的调用顺序做一些改变。

练习 4.5: 利用 `description` 环境, 制造一份会议参加人员及其地址的名单, 这里要把参加者的姓名作为 `\item` 命令的参数。

注意: 对于这三种列表的每一个而言, 若第一条 `\item` 命令前面有文本, 在处理时都会生成一个错误信息。

### §4.3.5 改变标签样式

在 `itemize` 和 `enumerate` 环境中的标签可以很容易地改变, 方法就是利用 `\item` 的可省参数。利用 `\item[+]`, 那么标签就变为 +, 而 `\item[2.1:]`, 标签就成为 2.1:。这里可省参数要取代标准标签。对于 `enumerate` 环境, 这就意味着对应的记数器并没有自动增 1, 用户必须手工改变它。

可省标签是在专为标签保留的区域中右对齐。该区域的宽度是该层次的缩进总长减去标签与正文的间距; 这意味着标签区域的左边界与包围它的外层左边界是对齐的。

可以改变文档中的全部或部分标准标签。在  $\text{\LaTeX}$  中是用如下内部命令给 `itemize` 生成标签的:

```
\labelitemi, \labelitemii, \labelitemiii, \labelitemiv
```

而相应于 `enumerate` 的是:

```
\labelenumi, \labelenumii, \labelenumiii, \labelenumiv
```

这里的 `i`, `ii`, `iii` 和 `iv` 分别指的是四个可能的层次。

可以用 `\renewcommand` 来改变这些命令。例如, 要把第三层的 `itemize` 标签从 \* 改为 +, 可以用

```
\renewcommand{\labelitemiii}{+}
```

同样, `enumerate` 环境中的标准标签也可以改变。然而, 这里还要复杂一些, 因为每个 `enumerate` 层次, 都还有一个记数器, 名字叫 `enumi`, `enumii`, `enumiii` 和 `enumiv`。一个记数器的值可以用命令 `\arabic`, `\roman`, `\Roman`, `\alph` 或 `\Alph` 来显示出来, 这些命令的意义从名字上就很容易知道了。即 `\Roman{xyz}` 用大写罗马数字显示出记数器 `xyz` 的当前值, 而 `\alph{xyz}` 则用的是小写字母 (这样 `a` 对应于 1, `z` 对应于 26)。

这些记数器, 同记数器样式命令一起, 可以用来重定义标签命令。例如, 要把第二层的标签改为阿拉伯数字后加 ‘.’, 那就需要用

```
\renewcommand{\labelenumii}{\arabic{enumii}.}
```

这样就把 `\labelenumii` 重定义为用阿拉伯数字显示的计数器 `enumii` 的值，后跟 ‘.’。用这种方法，所有的层次的编号都可以改变。而且甚至可以包含不至一个计数器：

```
\renewcommand{\labelenumii}{\Alph{enumi}.\arabic{enumii}}
```

这样就会使得每当在第二层调用 `\item`，那就会用作为大写字母的计数器 `enumi` 中的值，后跟数字形式的 `enumii` 中的值，即形式为：A.1, A.2, ..., B.1, B.2, ..., 等等。

如果要把新的标准标签适用于整个文档，那就应当把重定义命令放在导言中。否则，它就只对其所出现的那个环境内有效。

**练习 4.6:** 把第一层的 `itemize` 标准标签改为长破折号 —（写作 ---），而第二层的为 -（--），第三层只是一个连字符 -。

**练习 4.7:** 修改 `enumerate` 环境的标准标签，第一层为 (I), (II), ...，而第二层的形式为 I-1, I-2, ...，即第一层的大写罗马数字形式后跟第二层的数字形式的编号。

上面的练习表明 `itemize` 和 `enumerate` 环境中的标签可以改变为任何所期望的样式。然而，这种改变是不应给以鼓励的，因为从排版的角度来看，由 Leslie Lamport 所选定的标准标签是很难再做什么改进的。如果确实需要其它的设置，那应该整个文档中都应用这个设置。

### §4.3.6 参考文献

科技出版物中通常都会包含一长串引用，或者称为参考文献，它由其它工作的名称组成，在正文中通过其活动编号对其进行引用。通常正文没有结束，参考文献也就不会完成。

每当向参考文献中加入一项，如果需要通读所有的正文，以改变所有的引用编号，那是一件非常令人头痛的事。因此  $\text{\LaTeX}$  提供一种新的机制，它不但对参考文献进行格式化，而且跟踪对它进行的修改和填加，以在正文中自动改变引用。

参考文献可以用如下环境来生成：

```
\begin{thebibliography}{标签样本}
```

所有的参考文献项

```
\end{thebibliography}
```

在参考文献中和每一项都是用如下命令开始的：

```
\bibitem[标签]{关键词} 文本条目
```

如果没有可省参数 `标签`，`\bibitem` 就会生成一个位于中括号内的活动编号，以供在正文中引用。如果想有 `标签`，那你可以用任何东西做标签，如作者姓名的缩写，或者一个随意的引用编号。不可省参数 `关键词` 是不会出现

在参考文献中的引用关键词，而且将来在正文要被标签取代。关键词可以是字母，数字和除了逗号外的符号组成。

真正的参考文献信息是包含在 文本条目 中的，其形式可能为‘作者，题目，出版社，年代，版本，页码’，而且不同部分字样可能不同。第一行后的文本要进行缩进，宽度等于 标签样本 的宽度，因此标签样本应是参考文献中的最长标签。对于活动编号的标准应用，标签样本 应该是一个没有意义的数字，但其位数等于最大标签的位数（例如如果有超过 10 个，但不足 100 个标签，可以用 99 ）。

在正文中的引用由如下命令生成：

`\cite{ 关键词 }`

这里的 关键词 就是出现在 `\bibitem` 命令中的引用关键词。例如，如果参考文献中包含：

```
\begin{thebibliography}{99}
  \bibitem{lamport} Leslie Lamport. \textsl{\LaTeX\ -- A Document
    Preparation System}. Addison--Wesley Co., Inc.,
    Reading, MA, 1985
    . . . . .
  \bibitem{knuth} Donald E. Knuth. \textsl{Computers and
    Typesetting Vol. \A--E}. Addison--Wesley Co., Inc.,
    Reading, MA, 1984--1986
  \bibitem[6a]{knuth:a} Vol A: \textsl{The \TeX book}, 1984
  \bibitem[6b]{knuth:b} Vol B: \textsl{\TeX: The Program.}, 1986
    . . . . .
\end{thebibliography}
```

那么文本

For additional information about `\LaTeX\` and `\TeX\` see  
`\cite{lamport}` and `\cite{knuth, knuth:a}`.

的结果为：For additional information about  $\LaTeX$  and  $\TeX$  see [1] and [6,6a].

这里 lamport, knuth 和 knuth:a 被选作关键词。用 99 做标签样本，因为对于这里的 `\bibitem` 标准形式，两位数的标签就生成足够的缩进。关键词为 knuth 的项是列表中的第六项，因此自动得到标签 [6]；为了使其子项 knuth:a... 的编号为 [6a]... [6e]，因此需要把可省的标签参数设置为 6a, ..., 6e。

`thebibliography` 环境的结果是以参考文献的形式列于文档尾部。对于文档类 `book` 和 `report`，其开头会显示一个单词 **Bibliography** 做为章标题，而对于 `article`，则会显示 **References** 做为节标题。上面的那个示例参考文献结果为：

[1] Leslie Lamport.  *$\LaTeX$  – A Document Preparation System*. Addison–Wesley

Co., Inc., Reading, MA, 1985

.....

[6] Donald E. Knuth. *Computers and Typesetting Vol. A-E*. Addison-Wesley Co., Inc., Reading, MA, 1984-1986

[6a] Vol A: *The T<sub>E</sub>Xbook*, 1984

[6b] Vol B: *T<sub>E</sub>X: The Program.*, 1986

.....

本书实际所用的参考文献样式来自于 L<sup>A</sup>T<sub>E</sub>X 标准：在正文中的引用用的是作者姓名与出版日期。列于参考文献中所有项都没有标签。在 C.3.4 节中描述了如何生成这种样式的参考文献。

**练习 4.8:** 利用 `thebibliography` 环境生成一个参考文献，其标签由作者姓名前三个字母后接出版年代后两位数字组成。如果同一位作者在同一年里有不只一篇作品，那就增加一个活动字母以示区别，如 `knu86c`，`knu86d`。对于这样的标签，其同时也适合于做关键词。而缩进宽度应为 1.5cm。

**注意：**缩进宽度是由 `thebibliography` 环境中的标签样本参数确定的。这通常只是一个虚文本，所要的只是其宽度。这个宽度可由 `\hspace{ 宽度}` 精确给出。

**练习 4.9:** 把上面练习中的 `thebibliography` 环境复制到你一直在用的练习文件尾部（但要在 `\end{document}` 命令前面）。在正文中插入 `\cite` 命令来引用参考文献中的项。注意要保证 `\cite` 命令中的关键词与 `thebibliography` 环境 `\bibitem` 命令中的关键词完全一样。

通常 L<sup>A</sup>T<sub>E</sub>X 安装中也包含了程序 `BIBTEX`，它可以从 `\cite`（和 `\nocite`）命令，通过引用一个或多个参考文献数据库来自动生成参考文献。在 8.3.2 节和附录 B 中对此有详细介绍。后者也介绍了许多用户所用的参考文献数据库是如何构成的。

## §4.4 广义列表

类似于三种环境 `itemize`, `enumerate` 和 `description` 这样的列表可以从相当一般的形式构造而来。标签的类型与宽度，缩进的宽度，段落和标签之间的距离等等，都可以由用户通过 `list` 环境来全部或部分地进行设置：

```
\begin{list}{ 标准标签 }{ 列表声明 } 列表中的项 \end{list}
```

这里的 列表中的项 由列表的各项文本组成，每一项由 `\item` 命令开头，它要生成相应的标签。

标准标签 包含内容是当 `\item` 命令没有参数时生成的标签（见后）。

在 4.4.2 节中描述的列表参数可以由用户通过 列表声明 设为任何所期望的值。

## §4.4.1 标准标签

在 `list` 环境中的第一个参数是 标准标签，当 `\item` 命令没有参数时，要用它生成标签。对于无变化的标签，如 `itemize` 环境中的标签，这只要是所期望的符号就可以了。如果希望它是一个数符号，那就必须以 `$` 符号名 `$` 形式给出，即包含在 `$` 符号内。例如，要用  $\Rightarrow$  做标准标签，标准标签就必须定义为 `$\Rightarrow$`。

然而，通常标签中需要包含一系列数字。为此，就必须用 `\newcounter{名称}` 命令生成一个新的计数器，这里的 名称 就是它的标号。这条命令必须位于 `list` 环境中对这个计数器第一次应用之前。假设已为此而定义了一个计数器 `marker`，那么参数 标准标签 就可以用 4.3.5 节中所给出的任一命令来显示计数器：例如，`\arabic{marker}` 就生成一个活动的阿拉伯数字编号。

即使再复杂的标签也可以用这种方法得到。如果接连的编号为 A-I, A-II, ..., 标准标签 就要设为 `A--\Roman{marker}`。

一个计数器在 标准标签 里可以正常使用之前，必须通过在 列表声明 中包含命令 `\usecounter{计数器}` 来把该计数器与列表关联起来，这里的 计数器 就是赋给计数器的名称（上面例子中的 `marker`）。

标准标签实际上是由 `\item` 命令调用 `\makelabel{标签}` 得到的。用户可以借助于 `\renewcommand` 命令在列表中重定义 `\makelabel`：

```
\renewcommand{\makelabel}{新的定义}
```

如果标准标签是以这种方式定义的，在 `list` 环境中相应的项左边是空的。这是因为 `\makelabel` 是更一般的命令，它覆盖了其它定义。在 7.5.9 节中有例子。

## §4.4.2 列表样式参数

有很多样式参数，以格式化列表， $\text{\LaTeX}$  把它们都设置为特定的标准值。在特定的列表中，用户可以在 列表声明 中改变这些值。用 `\setlength` 命令可以像通常那样赋值。然而，如果这种赋值是在 `list` 环境外进行的，在大多数情形中，就简单地被忽略了。这是因为在每一层每一个参数都被预设置为默认值，只有 列表声明 才可以覆盖它。

下面列出了样式参数，同时在图 4.1 中进行了标识，它们是基于 Lamport 的选择：

```
\topsep
```

是一个竖直距离，其要与 `\parskip` 一起插入到列表与包围它们的上下文之间。其默认值在每一个列表层次中进行设置，不能在 列表声明 外面进行全局重定义。

```
\partopsep
```

当第一个 `\item` 项前面有空行，或者最后一个 `\item` 项后面有空行时，





**\labelsep**

是标签盒子与列表文本之间的距离。可以全局性地赋给其一个新值，但它只对第一层有作用。

**\itemindent**

是一个 **\item** 项中标签及文本第一行相对于右边界的缩进宽度。通常取为 0pt，因此就没有这种效果。只可以在 列表声明 中改变其值。

把当竖直距离从其标准值变为其它值，最好使用橡皮长度（2.4.2 节）。

由 **\item** 命令生成的标签通常在一个宽度为 **\labelwidth** 的盒子中是右对齐的。也可以如下面参数列表那样，使其为左对齐的，即在标准标签的定义或者 **\makelabel** 命令尾部加上 **\hfill**。

## §4.4.3 用户制作列表的示例

一个关于图的列表为：

**Figure 1:** *Page format with head, body, and foot, showing the meaning of the various elements involved.*

**Figure 2:** *Format of a general list showing its elements.*

**Figure 3:** *A demonstration of some of the possibilities for drawing pictures with  $\text{\LaTeX}$ .*

这个列表是用如下输入生成的：

```
\newcounter{fig}
\begin{list}{\bfseries\upshape Figure \arabic{fig}:}
  {\usecounter{fig}
  \setlength{\labelwidth}{2cm}\setlength{\leftmargin}{2.6cm}
  \setlength{\labelsep}{0.5cm}\setlength{\rightmargin}{1cm}
  \setlength{\parsep}{0.5ex plus0.2ex minus0.1ex}
  \setlength{\itemsep}{0ex plus0.2ex} \slshape}
  \item Page format with head, body, and foot, showing the
    meaning of the various elements involved.
  \item Format of a general list showing its elements.
  \item A demonstration of some of the possibilities for
    drawing pictures with \LaTeX.
\end{list}
```

命令 **\newcounter{fig}** 建立了一个新的计数器 **fig**。标准标签设置为直立的黑体单词 **Figure**，后接活动的阿拉伯数字，由：结尾。每个 **\item** 命令都会显示出一个这样的标签。

列表声明把 **\usecounter{fig}** 做为其第一条命令，这样就使得计数器 **fig** 在列表中可用。标签盒子的宽度（**\labelwidth**）设为 2.0cm，列表文本的

左边界 (`\leftmargin`) 为 2.6cm, 在标签和文本之间的距离 (`\labelsep`) 为 0.5cm, 列表的右边界 (`\rightmargin`) 离包围它的文本距离设为 1cm。

在一项中段落之间的竖直距离 (`\parsep`) 为 0.5ex, 但还可以伸展 0.2ex, 或收缩 0.1ex。项与项之间的额外距离 (`\itemsep`) 为 0ex, 可以伸展 0.2ex。

对于所有其它的列表参数, 用的都是标准值。在列表声明中最后一条命令是 `\slshape`, 这把列表文本设为 *slanted* 字样。

注意: 在  $\text{\LaTeX 2}_\epsilon$  中, 如果在标签定义中没有使用 `\upshape`, 每一 `\item` 的文本就是 *slanted* 字样的, 如 **Figure 1**。在  $\text{\LaTeX 2.09}$  中, 就必须用 `\bf` 和 `\sl` 声明, 但是不可能得到一个黑体的 *slanted* 字样。

#### §4.4.4 做为新环境的列表

如果在一个文档中要几次使用同一类型的列表, 那么在列表环境中输入同样的 标准标签 和 列表声明 是一件非常麻烦的事。 $\text{\LaTeX}$  提供了一种方法, 可以把给定列表定义为具有自己名称的环境。这是用命令 `\newenvironment` 来做到的。

例如, 可以用名称 `figlist` 保存上面例子中的列表, 以后可通过该名称调用它:

```
\newenvironment{figlist}{\begin{list}
  {\bfseries\upshape Figure \arabic{fig}:}
  {\usecounter{fig} ... {0ex plus0.2ex}\slshape}}
{\end{list}}
```

这样就可以如下调用它了:

```
\begin{figlist} 列表项 \end{figlist}
```

其行为同预定义的列表一样。

**练习 4.10:** 用名称 `sample` 定义一个新的环境, 它生成一个列表, 每当调用 `\item` 时, 生成标签为 Sample A, Sample B, 等等。标签在一个宽度为 20mm 的盒子中左对齐, 标签盒子与列表文本之间的距离为 2mm, 总的左边界为 22mm。文本的右边界相对于包围文本向内移了 5mm。两项之间的额外竖直距离除了通常的段间距外还有 1ex plus0.5ex minus0.4ex。在一项中第二段以后的段落具有 1em 的缩进。通常的段落间距为 0ex, 可伸展到 0.5ex。

$\text{\LaTeX}$  自身就经常用列表环境来定义许多其它的结构。例如, `quote` 环境的定义为

```
\newenvironment{quote}{\begin{list}{}
  {\setlength{\rightmargin}{\leftmargin}}
  \item[]}{\end{list}}
```

因此该环境也就是一个列表, 其 `\rightmargin` 的值被设置为 `\leftmargin` 的当前值, 其默认值为 2.5em。列表本身只由一个 `\item` 项组成, 它的标签

为空白，在 `quote` 的定义中自动调用了 `\item[]` 以达此目的。

用同样的方式， $\text{\LaTeX}$  在内部以特殊 `list` 形式定义了 `quotation` 和 `verse` 环境。左边界以及与包围文本的垂直距离都取的是 `list` 环境的标准值，因此只有当标准值改变时，其值才会变化。

最后，我们给出一个可能的自定义特殊列表做为例子：

```
\newenvironment{lquote}{\begin{list}{}{\item[]}\end{list}}
```

这生成一个 `lquote` 环境，其除了进行相对于包围文本进行 `\leftmargin` 的缩进，右边界与通常文本对齐（因为 `\rightmargin` 的标准值为 `0pt`）外，再没有别的变化。

#### §4.4.5 平凡列表

在  $\text{\LaTeX}$  中还包含一个 `trivlist` 环境，其语法为

```
\begin{trivlist} 内部文本 \end{trivlist}
```

在这里没有标准标签和列表声明参数。它相当于一个列表，其标签是空的，`\leftmargin`、`\labelwidth` 和 `\itemindent` 都赋值 `0pt`，而 `\listparindent` 等于 `\parindent`，`\parsep` 等于 `\parskip`。

$\text{\LaTeX}$  用这一环境生成其它环境。例如，当调用 `center` 环境时，内部实际上调用的就是

```
\begin{trivlist} \centering \item[] 内部文本 \end{trivlist}
```

环境 `flushleft` 和 `flushright` 也是类似定义的。

#### §4.4.6 嵌套列表

列表也可以与其它列表环境 `itemize`、`enumerate` 和 `\description` 之间相互嵌套，最大深度为 6。在每一层，都要相对于前面高一级那层进行总量为 `\leftmargin` 的缩进。

前面已提到过，只有在导言中才可能利用声明改变 `list` 参数的有限几个标准值。但有一个例外，那就是不同嵌套层数中左边界的缩进量。在内部它是用参数 `\leftmarginn` 来设置的，这里的 *n* 表示 *i*、*ii*、*iii*、*iv*、*v* 或 *vi*。用户可以改变这些值；例如，用声明 `\setlength{\leftmarginiv}{12mm}` 可以使得第四层的左边界相对于第三层向右移了 12mm。这些声明必须放在 `list` 环境外面，也不能放在列表声明里。

在嵌套列表的每一层中，都要调用内部宏 `\@listn`（*n* 从 *i* 到 *vi*）。它使 `\leftmargin` 的值等于相应的 `\leftmarginn`，除非 `\leftmargin` 在 `list` 环境中显式地进行了声明。也就是说，在外部并不存在单个的 `\leftmargin` 标准值，而是 6 个不同的值。只有在 `list` 环境内部，参数 `\leftmargin` 才有意义。

### §4.5 定理型的声明

在科技文献中，经常需要如下结构：

**Theorem 1 (Balzano–Weierstrass)** *Every infinite set of bounded points process at least one maximum point.*

或者

**Axiom 4.1** *The natural numbers form a set  $S$  of distinct elements. For any two elements  $a, b$ , they are either identical,  $a = b$ , or different from one another,  $a \neq b$ .*

同样的结构除了这里的名称 Theorem 或 Axiom 外，有时还会有名称 Definition, Corollary, Declaration 或 Lemma。它们的共同特点就是都有一个关键词和一个活动编号，而且它们以黑体字样显示，而其它相应的文本是斜体。

当然，这些结果用户可以通过显式给出样式和编号来得到，但是如果在中间插入那种类型的一个新结构，那么用户就需要对后面的结构进行麻烦的重新编号。利用命令

`\newtheorem{结构类型}{结构标题}[其它计数器]`

TeX 就会自动跟踪编号的变化。这里的 结构类型 是用户给该结构指定的任意一个名称，而 结构标题 就是那个以黑体字样与活动编号列在一起的单词（如 **Theorem**）。如果没有可省参数 其它计数器，在整篇文档中是连续编号的。然而，如果在 其它计数器 中有一个已存在的计数器的名称，如 `chapter`，每当该计数器增加时，都会对这里的结构重新编号，而且它们可以列在一起，如上面的 **Axiom 4.1**。

预定义的结构可以用如下命令来调用：

`\begin{结构类型}[附加标题] 文本 \end{结构类型}`

这里会给必要的计数器增 1，以生成恰当的编号。上面的例子就是用如下输入得到的：

```
\newtheorem{theorem}{Theorem} \newtheorem{axiom}{Axiom}[chapter]
```

```
. . . . .
```

```
\begin{theorem}[Balzano--Weierstrass] Every .... \end{theorem}
```

```
\begin{axiom} The natural numbers form ..... \end{theorem}
```

可以省略的 附加标题 紧接在活动编号后以黑体字样显示在小括号 () 内。

有时候一个结构的编号并不是与自己有关，它会与其它结构一起编号。为此可以在定义中包含另一个可省参数：

`\newtheorem{结构类型}[编号来源]{结构名称}`

这里的 编号来源 是一个已存在的定理结构名称，它们共享同一个计数器。因此通过定义 `\newtheorem{subthrm}{theorem}[Sub-Theorem]`，`theorem` 和 `subthrm` 这两个结构的编号是同一个序列：**Theorem 1**, **Sub-Theorem 2**,

Sub-Theorem 3, Theorem 4, 依次类推。

## §4.6 制表位

### §4.6.1 基础知识

在打字机上可以在一行的任何位置设置制表位；这样当按一些 tab 键，打字头或者回车键就会跳到下一个制表位处。

在 L<sup>A</sup>T<sub>E</sub>X 的 `tabbing` 环境中也具有类似的情形：

`\begin{tabbing}` 文本行 `\end{tabbing}`

可以认为 tab 停留点按从左到右具有顺序编号。在 `tabbing` 环境开始，除了左边界（它称为第零个制表位）外，没有其它的停留点。在一行中可以用命令 `\=` 在任何地方设置制表位，而一行是用命令 `\\` 结束的：

Here is the `\=first` tab stop, followed by `\= the second\\`

这里把第一个制表位设在紧接单词 the 空白后面，而第二个是在单词 by 的后面。

当以这种方法设置好了制表位后，就可以在后续的文本行中从左边界开始，用命令 `\>` 跳到任一个制表位处。新行还是用通常的命令 `\\` 开始。

例如：		<code>\begin{tabbing}</code>
Type	Quality	<code>Type\quad\= Quality\quad\=</code>
Color	Price	<code>Color\quad\= Price\\[0.8ex]</code>
Paper	med.	<code>Paper \&gt; med. \&gt; white \&gt; low \\</code>
Leather	good	<code>Leather \&gt; good \&gt; brown \&gt; high\\</code>
Card	bad	<code>Card \&gt; bad \&gt; gray \&gt; med.</code>
		<code>\end{tabbing}</code>

### §4.6.2 样本行

有时设置制表位比较好的，或者必须的方法是利用不必显示出来的样本行。例如，样本行可以由后面出现的各列中最宽项组成，或者由制表位之间最小列间距组成。在样本行中也可以包含 `\hspace` 命令，这样可以保证制表位之间的距离大于预定义的长度。

为了不显示样本行，那么该行文本不是用 `\\` 结束，而用的是 `\kill`。

`\hspace*{3cm}\=sample column \=\hspace{4cm}\= \kill`

除左边界外，上例共设置了三个制表位：

在样本行开头的 `\hspace` 命令必须是 \*- 形式，否则就会去掉在行边界所插入的间距。

### §4.6.3 制表位与左边界

在 `tabbing` 环境中，每行的左边界首先要与包围该环境的文本左边界一致，并标记为第零个制表位。通过在一行的开头加上制表键 `\>`，可以使文本在第一个制表位处开始。然而，放在第一行开头处的命令 `\=` 会使得后续各行得到同样的效果。用 `\+\>` 开始或结束一行，会使得后面的行都是开始于两个制表位后。只要一行中可以用多少个制表位，就可以使用多少个 `\+` 命令。

而命令 `\-` 具有相反的效果：它把后续各行的左边界向左移动一个制表位。但用这条命令不能把边界设在第零个制表位的左边。

在每一行中要覆盖 `\+` 命令的效果，可以采用在要被去掉的制表位前面加上 `\<` 的方法。该行就拥有到左边界那么多的制表位。对于下一个 `\>` 命令，新行就在由 `\+` 和 `\-` 命令总数所确定的左边界处开始。

### §4.6.4 其它的制表命令

在任一行上都可以重设或增加制表位。如果具有足够的 `\>` 命令跳到一个制表位，用 `\=` 命令就会插入一个制表位，否则它会重设下一个制表位。

例如：

```
Old column 1 Old column 2      \begin{tabbing}
Left column  Middle col Extra col Old column 1 \= Old column 2\\
New col 1 New col 2      Old col 3 Left column \> Middle col
Column 1 Column 2      Column 3 \= Extra col\\
                                New col 1 \= New col 2 \>
                                Old col 3\\
                                Column 1\> Column 2 \> Column 3
                                \end{tabbing}
```

而有时候，在重设了制表位后又希望使用原来的。命令 `\pushtabs` 可以用来完成这一任务，它把当前制表位保存起来，并把它们从当前行上去掉。这样所有的制表位都可以重新设置。可以用命令 `\poptabs` 来激活保存的制表位。`\pushtabs` 可以应需要而使用任意次，但在同一个 `tabbing` 环境中必须有与它相同数目的 `\poptabs` 命令。

可以用 `左边文本\'` `右边文本` 来在一个制表位处定位文本，这里的 `左边文本` 指的是恰好位于当前制表位前面（或者左边界）且具有很小的距离；而 `右边文本` 恰好在制表位处开始。在 `左边文本` 与制表位之间的距离由制表参数 `\tabbingsep` 确定。用户可以用命令 `\setlength` 如通常那样来修改其值。

可以用命令 `\'文本` 来使文本相对于右边界对齐。在这一行中余下部分必须再没有 `\>` 或 `\=` 命令。

`\=`、`\'` 和 `\` 命令在 `tabbing` 环境外是重音命令（2.5.7 节）。如果在 `tabbing` 环境内需要这些重音，那就必须用 `\a=`、`\a'` 和 `\a` 来代替。例如，要在 `tabbing` 环境内生成 `ó`、`ò` 或 `ō`，就必须用 `\a'o`、`\a'o` 或 `\a=ō`。`\-`

命令在 `tabbing` 环境外也具有另外一种意义（单词的建议断点），但由于在这个环境中不会自动断行，因此不需要有替代形式。

下面是一个演示所有制表命令的例子：

Apples:	consumed by: people	<code>\begin{tabbing}</code>
	horses	<code>Graefruit: \= \kill</code>
	and sheep	<code>Apples: \&gt; consumed by: \= people\+ \+ \\\</code>
	reasonably juicy	<code>horses \\\</code>
Grapefruits:	a delicacy	<code>and \' sheep \- \\\</code>
(see also: melons		<code>reasonably juicy \- \\\</code>
pumpkins)		<code>Grapefruits: \&gt; a delicacy \\\</code>
Horses	feed on apples	<code>\pushtabs</code>
		<code>(see also: \= melons \\\</code>
		<code>\&gt; pumpkins) \\\</code>
		<code>\poptabs</code>
		<code>Horses \&gt; feed on \&gt; apples</code>
		<code>\end{tabbing}</code>

#### §4.6.5 关于制表的注解

$\text{\TeX}$  如通常处理段落那样处理 `tabbing` 环境，如果在环境中必要的话，会在两行中断开。然而，不允许在其中使用 `\newpage` 和 `\clearpage` 命令，而忽略 `\pagebreak` 命令。如果用户要强迫在 `tabbing` 环境中分页，那么他（或她）可以使用一个小技巧：在希望分页的那行末尾定义一个相当大的行间距（如 `\\[10cm]`）。这就会强迫在此处分而，而且在下一页上所定义的空白是不存在的。

在一对 `{ }` 中的文本行仍然有效，因此在某一行上的尺寸或字体声明只对该行有作用。文本不需显式地放在一对大括号内。

不可能在一个 `tabbing` 环境中嵌套另一个 `tabbing` 环境。

注意：制表移动命令 `\>` 总是移到下一个逻辑制表位处。如果前面的文本长度超过两个制表位之间可用的间距时，这实际上是一个回移。这与打字机上的制表机制不同。

在 `tabbing` 环境中没有自动的断行。在没有遇到 `\\` 命令之前，一行是不会断开的。这样文本就有可能延伸超过右边界。这就需要依靠用户来决定到底该怎么办了。

在 `tabbing` 环境中 `\hfill`、`\hrulefill` 和 `\dotfill` 命令是没有作用的，因为这里不会发生伸展。

**练习 4.11:** 利用 `tabbing` 环境生成下面这个表格。

Project:	Total Requirements =	\$900 000.00
of which	1995 =	\$450 000.00
	1996 =	\$350 000.00
	1997 =	\$100 000.00
1995 approved:	\$350 000.00	Deficiency: \$100 000.00

1996	\$300 000.00	\$150 000.00
1997	\$250 000.00 Surplus:	\$150 000.00
tentative	1996 = \$100 000.00 for deficiency 1995	
	1997 = \$ 50 000.00	1996
	+ \$100 000.00	excess for 1995 in 1996
Commitments	1995 = \$100 000.00	
	1996 = \$1 50 000.00	signed: H. André

提示：在 `tabbing` 环境中的第一行应该是

```
Project: \=Total Requirements\= = \$900\,000.00 \+\
```

该行末尾的 `\+` 有什么作用呢？为了使第二行到第四行上 1995、1996 和 1997 年都开始于第二个制表位，该如何操作呢？在第二行的 `\` 命令前应用哪个命令呢？

虽然第八行有额外的制表位，但第一行到第四行以及第八行到第十二行的制表位是一致的。利用 `\$1=00\,000.00`，可以使得第九行上的 `\$>50\,000.00` 项与上面所有行中的小数点对齐。

第 5-7 行具有自己的制表位。这里要使用制表位的保存与恢复功能，进行制表位重设，并在结束时返回原状态。第 5-7 行的左边界对应于前面的第一个制表位。那么为此在第四行末尾应该用哪条命令呢？如何重设第二个到最后一个制表位呢？

最后一行的 ‘signed: H. André’ 是右对齐的。在 `tabbing` 环境中是如何做到这点的呢？要小心处理在 `tabbing` 环境中的重音 `é`。

## §4.7 盒子

所谓 盒子，就是一部分文本，`TeX` 把它做一个整体，如同单个字符那样进行处理。一个盒子（连同其中的文本）可以上下左右移动。由于盒子是一个整体，`TeX` 再不能把它分开，即使最初它是由更小的不可分的盒子组成。然而，如果乐意的话，可以把更小的盒子放在一起，以构造一个更大的盒子。

这实际上也就是 `TeX` 内部进行格式化时的方法：单个字符构成字符盒子，然后把它放到水平的行盒子中，在单词间插入橡皮长度。行盒子竖直堆积，构成段落盒子，在行之间也要插入橡皮长度。然后把它们放到页面主体盒子中，其连同页眉和页脚盒子一起构成页面盒子。

在 `LaTeX` 中，用户可以选择三种类型的盒子：LR 盒子，段落盒子和标尺盒子。LR（左 - 右）盒子是由水平的从左到右的有序材料组成。段落盒子则是由竖直堆积的行组成。标尺盒子是一个用黑色填充的实心矩形，通常用来画水平或竖直线。



## §4.7.1 LR 盒子

要创建一个由 LR 模式中文本组成的盒子，可以用命令

`\mbox{ 文本 }` 和 `\makebox[ 宽度 ][ 位置 ]{ 文本 }`

`\fbox{ 文本 }` 和 `\framebox[ 宽度 ][ 位置 ]{ 文本 }`


左边两条命令生成一个 LR 盒子，其宽度恰好是在 `{ }` 中所给 文本 的宽度。`\fbox` 命令同 `\mbox` 命令一样，只是 文本 要用方框包围起来。

而对于右边两条命令，宽度是由可省的长度参数 宽度 来预先确定的。另一个可省参数 位置 定义 文本 在盒子的位置。如果不给任何值，文本 是居中排列的。位置 参数可以取如下值：


l 文本左对齐，

r 文本右对齐的，

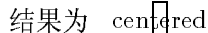
$2\varepsilon$  s 伸展文本，以达到所定义的宽度。


因此 `\makebox[3.5cm]{centered text}` 生成一个宽度为 3.5cm 的盒子，其中的文本是居中排列的，即 centered text，多余地方是用空白填充的，而 `\framebox[3.5cm][r]{right justified}` 的结果是文本在一个宽度为 3.5cm 的盒子中右对齐：。对于  $\text{\LaTeX 2}\varepsilon$ ，也可以用

`\framebox[3.5cm][s]{stretched\dotfill text}`

以生成一个填满的盒子：，在这种情形中需要利用橡皮长度或者其它的填充物（42 页）以填充伸展出现的空白。

如果 文本 的自然宽度要比所定义的 宽度 大，那就会视 位置 参数的不同值，而在盒子的左边、右边或两边突出。例如，

`\framebox[2mm]{centered}` 结果为 

上面这种用法对于 `\framebox`，确实显得有点儿愚笨，但是对于 `\makebox` 却可能非常有用。在 `picture` 环境的图示中，可以利用 `\makebox` 定义一个宽度为 0pt 的盒子，这样就可以生成一个居中的，或左（右）对齐的文本（见第 6 章的例子）。用它也可以使得两部分文本重叠，例如 `\makebox[0pt][l]{/}S` 的结果为一条斜线穿过 S，即 。

注意：长度的定义中必须包含长度单位，即使所定义的长度是零。因此这里定义零宽度必须用 `0pt`，而不能只用 `0`。

在  $\text{\LaTeX 2}\varepsilon$  中也可以相对于一个 LR 盒子的自然宽度（即只用命令 `\mbox` 时的宽度）来定义其 宽度：

$2\varepsilon$  `\width` 表示盒子的自然宽度，

$2\varepsilon$  `\height` 表示从基线到顶部的距离，

$2\varepsilon$  `\depth` 表示从基线到底部的距离，

$2\varepsilon$  `\totalheight` 就是 `\height` 加上 `\depth`。

要生成一个宽度为其中包含文本的总高度六倍的有框盒子，可以用命令：

```
\framebox[6\totalheight]{Text} Text
```

注意：这些特殊的长度参数只在一个 LR 盒子的 宽度 定义或者下面将要介绍的段落盒子的 高度 定义中才有意义。而出现在其它情形中，就会产生错误信息。

如果一部分文本要出现在文档的几个地方，那么可以把它们保存起来，首先用如下命令：

```
\newsavebox{\boxname}
```

这样就可以创建一个叫 ‘\boxname’ 的盒子。该名称要符合 L<sup>A</sup>T<sub>E</sub>X 命令语法（只能有字母），并且前面有 \。其也不能与任何已存在的 L<sup>A</sup>T<sub>E</sub>X 命令同名。一旦已经如此初始化了一个盒子，那么命令：

```
\sbox{\boxname}{文本} 或者
```

```
\savebox{\boxname}[宽度][位置]{文本}
```

就会把 文本 的内容保存起来，以备后面使用。这里的可以省略参数 宽度 和 位置 与 \makebox 和 \framebox 中的意义一样。接着就可以在需要用命令

```
\usebox{\boxname}
```

来把保存的文本当做一个整体插入文档中。

LR 盒子的内容也可以用如下环境来保存：

```
2ε \begin{lrbox}{\boxname}
  文本
\end{lrbox}
```

其等价于 \sbox{\boxname}{文本}。其好处在于它可以保存位于用户定义的环境（7.4 节）中的 文本，以备将来用 \usebox 插入。

### §4.7.2 LR 盒子的竖直移位

命令

```
\raisebox{上移量}[高度][深度]{文本}
```

生成一个内容为 文本 的 \mbox，它相对于当前基线向上移动了 上移量。可省参数告诉 L<sup>A</sup>T<sub>E</sub>X 该盒子在基线上方伸展的 高度，基线以下的 深度。若没有这些参数值，盒子具有由 文本 和 上移量 确定的自然尺寸。注意这里的 上移量、高度 和 深度 都是长度（2.4.1 节）。如果 上移量 为负数，那么盒子就相对于基线向下移动。

例如：

```
Baseline \raisebox{1ex}{high} and \raisebox{-1ex}{low}
and back again
```

结果为： Baseline <sup>high</sup> and <sub>low</sub> and back again。

高度 和 深度 的值可以与 文本 的实际值完全不同。其结果是根据同一行上所有其它盒子（字符也是盒子）的高度和深度，来确定当前文本行与前后

文本行的距离。当提升了一个盒子，而高度却是正常的字符大小，那么被提升的盒子就会与上面的文本行重叠，同样当降低一个盒子时深度也具有同样的效果。

### §4.7.3 子段盒子和小页

整个段落可以用如下命令来放到单独的竖直盒子（或者套用 L<sup>A</sup>T<sub>E</sub>X 术语称为 子段盒子）中：

```
\parbox[位置]{宽度}{文本}
```

也可以用环境实现同样的效果：

```
\begin{minipage}[位置]{宽度} 文本 \end{minipage}
```

其都生成一个给定宽度的竖直盒子，其中的文本行如通常段落模式一样彼此堆积。

可省的参数位置有可取如下值：

- b 盒子的底边与当前基线对齐，
- t 顶行文本与当前基线对齐。

当没有任何位置参数时，子段盒子相对于外部文本的基线竖直居中排列。

位置参数只有当 `\parbox` 命令或者 `minipage` 环境位于一个段落内部时才有意义，否则当前行与基线都没有意义。如果子段盒子前面紧接一个空行，这就是要开始一个新段。在这种情形中，子段盒子的竖直位置是由段落中后面的元素确定的。而后面的元素也可以还是子段盒子。如果整个段落只是由一个子段盒子或者小页组成，位置参数也没有意义，是无效的。

例：

```
\parbox{3.5cm}{\sloppy This is a 3.5 cm wide parbox. It is
vertically centered on the}
\hfill CURRENT LINE \hfill
\parbox{5.5cm}{Narrow pages are hard to format. They usually
produce many warning messages on the terminal. The command
{\tt\symbol{92}sloppy} can stop this.}
This is a 3.5 cm wide
parbox. It is vertically centered on the
Narrow pages are hard to format.
They usually produce many warn-
ing messages on the terminal. The
command \sloppy can stop this.
\begin{minipage}[b]{4.3cm}
The minipage environment creates a vertical box like the parbox
command. The bottom line of this minipage is aligned with the
\end{minipage}\hfill
\parbox{3.0cm}{middle of this narrow parbox, which in turn is
aligned with}
\hfill
\begin{minipage}[t]{3.8cm}
the top line of the right hand minipage. It is recommended that
the user experiment with the positioning arguments to get used
to their effects.
\end{minipage}
```

The `minipage` environment creates a vertical box like the `parbox` command. The bottom line of this `minipage` is aligned with the middle of this narrow `parbox`, which in turn is aligned with the top line of the right hand `minipage`. It is recommended that the user experiment with the positioning arguments to get used to their effects.

在 4.7.7 节中我们演示了如何按所希望的那样把子段盒子彼此竖直堆积起来。

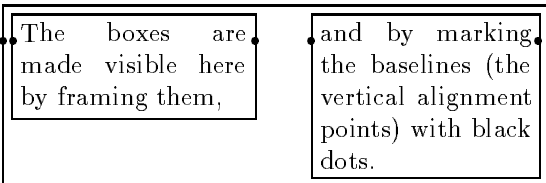
`\parbox` 命令生成一个由文本组成的竖直盒子，这同 `minipage` 环境是一样的。但是后者更具有一般性。在 `\parbox` 中的文本不可以包含在 4.2–4.5 节中所讲解的居中、列表或者其它环境。而另一方面，在 `minipage` 环境中是完全可以包含这些环境的。也就是说，一个 `minipage` 可以包含居中或缩进文本，也可以有列表和制表。

#### §4.7.4 竖直摆放的问题

对小页或者子段盒子的竖直定位有时会产生意想不到的结果，我们可以用图形来形象地说明  $\text{\LaTeX}$  是如何处理盒子的。假设我们想把两个高度不同的盒子并排摆放，并且对齐第一行，而它们的底部与当前文本行对齐。‘最明显的’做法就是

```
\begin{minipage}[b]{...}
  \parbox[t]{...}{...} \hfill \parbox[t]{...}{...}
\end{minipage}
```

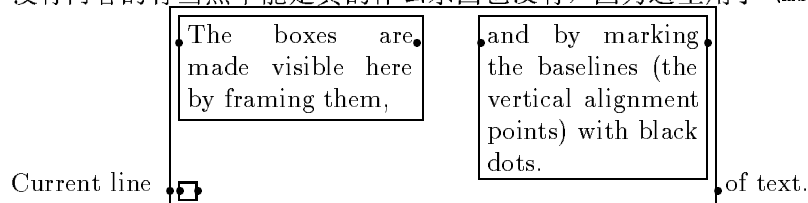
但这行不通，因为它的结果是：

Current line  of text.

之所以如此的原因在于  $\text{\LaTeX}$  内部把每个子段盒子或小页都当做一个具有自己相对于基线上下的高度和深度的字符来对待。当要考虑最外层的小页时，它只是在同一行上有两个‘字符’，而且该行同时是顶行和底行。因此最外面的小页的底行确实与文本行对齐，但它同时也是顶行。解决的办法是在最外层盒子中加一个没有内容的第二行：

```
\parbox[t]{...}{...} \hfill \parbox[t]{...}{...} \\ \mbox{}
```

没有内容的行当然不能是真的什么东西也没有，因为这里用了 `\mbox`。



如果要对齐两个盒子的底行，而顶部与当前行对齐，也会有同样的问题出现。下面是两种加入没有内容的第一行的方法：

`\mbox{} \` 恰好顶部对齐，或者

`\mbox{} \[-\baselineskip]` 第一行文本对齐。

在 161 页上有第一种情形的一个例子。在第 81 页的练习 4.12 中也需要用到没有内容的行。

#### §4.7.5 具有指定高度的段落盒子 2ε

在  $\text{\LaTeX} 2_{\epsilon}$  中，`\parbox` 命令和 `minipage` 环境的语法已进行了推广，可以还有两个可省参数：

2ε `\parbox[位置][高度][内部位置]{宽度}{文本}`

2ε `\begin{minipage}[位置][高度][内部位置]{宽度}`

文本

`\end{minipage}`

在这两种情形中，高度定义盒子的高度；在高度参数中，可以同 `\makebox` 和 `\framebox` (75 页) 中的宽度参数一样，使用 `\height`, `\width`, `\depth` 和 `\totalheight` 参数。

可省参数 内部位置 说明文本在内部是如何定位的，其只有当给定了高度时才有意义。可能的取值为

- t 要文本靠盒子的顶部，
- b 把文本推向盒子的底部，
- c 竖直居中，
- s 伸展文本以填满整个盒子。

在最后那种情形中，当出现竖直伸展时，要用到橡皮长度 (2.4.2 节)。

注意外部定位参数 位置 与内部定位参数 内部位置 的区别：前者说的是盒子如何与包围它的文本对齐，而后者决定在盒子中的内容是如何安排的。

例：

```
\begin{minipage}[t][2cm][t]{3cm}
```

```
  This is a minipage of height 2~cm with the text
  at the top.
```

```
\end{minipage}\hrulefill
```

```
\parbox[t][2cm][c]{3cm}{In this parbox, the text
```

```

    is centered on the same height.}\hrulefill
\begin{minipage}[t][2cm][b]{3cm}
    In this third paragraph box, the text is at the bottom.
\end{minipage}

    This is a minipage_____
    of height 2 cm with          In this parbox, the
    the text at the top.         text is centered on
                                the same height.
                                In this third para-
                                graph box, the text
                                is at the bottom.


```

盒子间的 `\hrulefill` 命令显示了基线的位置。这三个盒子的尺寸是一样的，区别只是内部位置的值不一样。

#### §4.7.6 标尺盒子



所谓标尺盒子就是完全用黑色填充的矩形。其相应命令的一般语法为

```
\rule[提升]{宽度}{高度}
```

其生成一个具有给定宽度，高度的实心盒子，并且相对于基线向上做提升。因此 `\rule{8mm}{3mm}` 的结果是 。若没有可省参数提升，矩形就放在当前文本行的基线上。

参数提升、宽度和高度都是长度（2.4.1 节）。如果提升为负数，则矩形位于基线以下。

也可以生成一个宽度为零的标尺。这样就得到一个不可见的具有给定高度的盒子。如此的构造称为一个支撑，这样可以强迫其所在的水平盒子具有不同于其所包含内容需要的高度和深度。对于这种情形，用 `\vspace` 就行不通，因为它只是增加已经存在的竖直间距。

例如，`\fbox{Text}` 的结果为 。为了得到 ，那就必须告诉  $\text{\TeX}$ ，盒子中的内容要相对于基线向上和向下延伸一定的长度。这可以用 `\fbox{\rule[-2mm]{0cm}{6mm}Text}` 来实现。这也就是说要用方框包围起来的文本是‘一个不可见的柱子，其底位于基线下 2mm，高有 6mm，它后面接单词 Text’。这个竖直条虽然看不到，但它却可以确定方框的顶边和底边。

标尺盒子的高度也可以是零，这就是一个不可见的具有给定宽度的水平线；然而，这种构造应是没有什么实际作用的，因为可以用 `\hspace` 命令很容易得到水平位移。

#### §4.7.7 嵌套盒子

上面描述的盒子可以相互嵌套任意次。要把一个 LR 盒子包含在一个子段盒子或者小页中没有任何概念上明显困难。相反，把一个子段盒子放在 LR 盒子中也是可能的，只要头脑中具有这样的想法：每个盒子都是一个整体， $\text{\TeX}$  把它们当做具有相应尺寸的单个字符一样来处理，那么一切就很清楚了。

把子段盒子放在 `\fbox` 命令内部，可以使得整个子段盒子都用方框包围起来。当前的结构就是用

```
\fbox{\fbox{\parbox{10cm}{把子段盒子...  }}}
```

得到的一个在方框盒子内的宽度 10cm 的子段盒子，而这个方框盒子又位于另一个方框盒子的内部，这样可以得到双层方框的效果。

把子段盒子包围在 `\raisebox` 内，可以得到任意希望的竖直位移。这里的两个盒子都具有位置参数 `[b]`，而右边的那个是如下生成的：

```
\raisebox{0.5cm}{\begin{minipage}[b]{2.5cm}
  a b c d e ... x y z\\
  \underline{baseline}
\end{minipage} }
```

```
a b c d e f g h i
j k l m n o p q r
s t u v w x y z
baseline
```

它向上位移了 1cm。  
baseline

一个很有用的结构就是在一个 `minipage` 环境中，两个 `minipage` 环境相互定位。外层 `minipage` 的位置参数可以用来把其内容做为一个整体，与相邻文本或者盒子对齐。在练习 4.12 中给出了如此结构的一个例子。

最后提一下，类似 `\parbox` 命令和 `minipage` 环境这样的竖直盒子也可以做为 `\sbox` 或者 `\savebox` 命令中的文本，从而保存起来，以后可以通过 `\usebox` 调用它们，具体与 4.7.1 节中的描述一样。

#### §4.7.8 盒子样式参数

对于有框盒子 `\fbox` 和 `\framebox`，有两个用户可以重设的样式参数：

`\fboxrule` 确定方框线的粗细，

`\fboxsep` 设置方框与被包围文本之间的距离大小。

与通常的  $\text{\LaTeX}$  方式一样，用 `\setlength` 命令可给这些长度参数赋新值：利用命令 `\setlength{\fboxrule}{0.5mm}`，可以使得所有后面的 `\framebox` 和 `\fbox` 命令中的线粗都为 0.5mm。

这些设置的适用范围也遵从通常的规则：如果其位于导言中，那么其适用于整个文档；如果它位于一个环境中，那么其有效性持续到环境结束。

这些参数对于用在 `picture` 环境（6.4.2 节）中的 `\framebox` 命令没有作用。因为那里面的 `\framebox` 命令的语法和作用要比通常时候的更加广泛。

**练习 4.12:** 如何生成下面这个嵌套结构？（注意：字体尺寸是 `\footnotesize`）

<p>The first line of this 3.5cm wide minipage or parbox is aligned with the first line of the neighboring minipage or parbox.</p>	<p>This 4.5cm wide minipage or parbox is positioned so that its top line is at the same level as that of the box on the left, while its bottom line is even with that of the box on the right. The naive notion that this arrangement may be achieved with the positioning arguments set to <code>t</code>, <code>t</code>, and <code>b</code> is incorrect. Why? What should this selection really produce?</p>	<p>The true solution involves the nesting of two of the three structures in an enclosing minipage, which is then separately aligned with the third one.</p>
---	--	---

注意：由于有可能左边和中间的结构首先包围在小页中，或者是中间与右边的结构包围在小页，因此可以有两种不同的解法。试给出这两种解法。顺带提一下，第三个小页的宽度为 3cm。

注意：这里又出现了两个盒子并排放在一起，如何与同行文本对齐的问题，（4.7.4 节）。为了使得正确地竖直对齐，需要加入一个没有内容的行。

**练习 4.13:** 生成下面给出的那个有框结构，并把它用命令 `\sbox{\warning}{structure}` 保存起来。为此你需要先用 `\newsavebox{\warning}` 创建一个叫 `\warning` 的盒子。这样以后你就可以在练习文件中任何需要打印这条警告信息的地方用 `\usebox{\warning}`。

Vertical placement of minipage and parboxes can lead to surprising results which may be corrected by the use of dummy lines.

注意：这个子段盒子的宽度是 10cm。如果依照前面有双层框的例子一样做，要生成这个有框文本，应该没有什么困难。要确保在 `\sbox{\warning}{结构}` 的最后输入了正确数目的右大括号。

然后，改变方框线粗细（`\fboxrule`）和方框间距（`\fboxsep`）的值，并再次打印出结果。

## §4.8 表格

利用前面几节中的 盒子 结构和 `tabbing` 环境，可以生成各种类型的有框或无框表格。然而，`TeX`还提供了更方便的建立如此复杂结构的方法。

### §4.8.1 构造表格

环境 `tabular`, `tabular*` 和 `array` 是生成表格和矩阵的基本工具。这些环境的语法如下：

<code>\begin{array}[位置]{列}</code>	行	<code>\end{array}</code>
<code>\begin{tabular}[位置]{列}</code>	行	<code>\end{tabular}</code>
<code>\begin{tabular*}[宽度][位置]{列}</code>	行	<code>\end{tabular*}</code>



`array` 环境只能用在 数学模式 (见第 5 章) 中。只所以在这里提及它, 就是因为它的语法和参数意义与 `tabular` 环境中的完全一样。这三种环境都创建一个子段。参数意义如下:

**位置** 竖直定位参数 (也可以看 4.7.3 节子段盒子中同名参数的意义)。它可以取下列值:

- `t` 表格顶部与当前外部文本行的基线对齐;
- `b` 表格底部与外部基线对齐;

当没有定位参数时, 表格相对于外部基线居中摆放。

**宽度** 该参数只能出现在 `tabular*` 环境中, 其确定它的整体宽度。在这种情形中, `列` 参数必须在第一项后面某个地方包含 `@-` 表达式 (细节见下) `@{\extracolsep{\fill}}`。对于其它两种环境, 整体宽度是由其文本内容确定的。

**列** 列格式参数。除了可能存在的相应于表格左右边界和列间距的额外项外, 每列都必须在其中有一个相应的项。

可能的列格式符号有

- `l` 列内容是左对齐的;
- `r` 列内容是右对齐的;
- `c` 列内容是居中的;

`p{宽}` 在该列的文本设置成具有给定 宽 的行, 顶行与其它列对齐。实际上文本是用命令 `\parbox[t]{宽}{列文本}` 放在一个子段盒子中的;

`*{数}{列}` 包含在 列 中的列格式被复制了 数 份, 因此 `*{5}{|c|}` 的结果与 `|c|c|c|c|c|` 相同。

相应于左右边界和列间距的可用 格式化符号 有:

- `|` 画一条竖直线;
- `||` 画两条紧相邻的竖线;

`@{文本}` 这一条也叫做 `@-` 表达式, 它在自己出现的两列中间每一行上插入 文本 。

`@-` 表达式去掉了原本自动加在两列之间的空白。如果在插入文本和后面的列之间需要有空白, 那么需要在 `@-` 表达式的 文本 中包含 `\hspace{ }` 命令。如果想使某两个特定列之间的距离与其它的标准间距有所不同, 那么可以通过在格式参数中对应于这两列的地方放上 `@{\hspace{宽}}`, 这样就可能很容易地达到目标了。这里给定宽度的空白取代了标准列间距。

在 `@-` 表达式中的 `\extracolsep{宽}` 会使得所有后面的列间距都增加给定宽度的额外间距, 其作用持续到下一个 `\extracolsep` 为止。与标准间距不同, 后面的 `@-` 表达式并不会去掉这个额外空白。在

`tabular*` 环境中, 在列格式中的某处必须有 `@{\extracolsep\fill}` 命令, 以使得后面所有列间距可以伸展到预定义的表格宽度。

如果表格的左右边界并没有竖线, 那么就会在该处加入等于通常列间距一半的空白。如果不希望加入这个空白, 可以在列格式的开始或结尾处包含一个空的 `@-` 表达式 `@{ }`, 以删掉这种空白。

行 由表格的实际条目组成, 每一水平行都由 `\\` 结束。这些行是由一组彼此之间用 `&` 符号分开的列条目组成。因此每一行应具有与在列定义 列 中相同数目的列条目。可以有些条目是空白的。TeX 把每个列条目当做就好像用大括号 `{ }` 包围起来一样, 因此对于类型样式或尺寸的修改, 将被局限在这个列中。

`\hline` 这条命令只能位于第一行前面, 或者紧接在行结束符 `\\` 后面。它在刚结束的那行下面画一条水平直线, 或者如果其位于开头时, 在表格顶部画一横线, 横线的宽度与表格的宽度相同。

放在一起的两条 `\hline` 命令就会画出两条间隔很小的水平线。

`\cline{n-m}` 该命令从第  $n$  列的左边开始, 画一条到第  $m$  列右边结束的水平线。与 `\hline` 一样, 它也只能位于行结束符 `\\` 的后面, 而且可以同时有多次。命令 `\cline{1-3}` `\cline{5-7}` 就会在刚结束的行下面画两条水平线, 一条是从第 1 列到第 3 列, 另一条是从第 5 列到第 7 列。在每种情形下, 用的都是完全列宽。

`\multicolumn{数}{列}{文本}` 该命令把接下来的 数 列组合成单个列, 其宽度等于总宽度加上列间距。参数 列 由一个定位参数 `l`, `r` 或 `c`, 以及可能有的 `@-` 表达式和竖线 `|` 组成。通过把 数 的值取 1, 可以改变特定行中某一列的定位参数。

在这种情况下, ‘列’ 是由定位符号 `l`, `r` 或者 `c` 开始的, 包含所有接下来的内容, 直到遇到另一个定位符号为止。第一列也可以包含在第一个定位符号之前的内容。因此 `|c@{}r|` 就包含三列: 第一列是 `|c@{ }`, 第二列是 `r`, 第三列是 `r|`。

`\multicom` 命令只能位于一行的开始或者一个列分隔符 `&` 后面。

`\vline` 该命令画一条竖直线, 其高度等于其所位于地方的行高。用这种方法, 可以得到那些不是贯穿整个表格的竖直线。

对于 `tabular` 和 `array` 环境, 也可以用 L<sup>A</sup>T<sub>E</sub>X 2<sub>ε</sub> 命令 `\tabularnewline` 结束一行。这是一条明确的行结束符, 而 `\\` 可以在一个列条目中结束一个文本行。(该命令是在 1994 年 12 月 1 日引进的。)

由于表格是与子段盒子和小页一样的竖直盒子, 因此它也可以与其它的盒子或者文本水平定位 (见 4.7.3 节的例子)。特别要指出的是, 为了使表格

在页面上居中表格必须包围在

```
\begin{center} 表格 \end{center}
```

### §4.8.2 表格样式参数

在表格的生成中， $\text{\LaTeX}$ 要利用许多样式参数，来设置其标准值。用户也可以改变这些值，既可以在导言中进行全局性改动，也可以只局限于一个环境中。但不能在表格内部对其进行改动。

`\tabcolsep` 是插入在 `tabular` 和 `tabular*` 环境中两列间距离的一半；

`\arraycolsep` 是在 `array` 环境中相应于列间距的一半；

`\arrayrulewidth` 是表格中水平线与竖直线的粗细；

`\doublerulesep` 是双直线时两线之间的距离。

可以用 `\setlength` 命令像通常那样改变这些参数的值。例如，利用命令 `\setlength{\arrayrulewidth}{0.5mm}` 可使直线粗变为 0.5mm。除了上面的参数外，还有参数

`\arraystretch` 可以用来修改表格中的行间距。这是一个放缩因子，标准值为 1。取值 1.5 就意味着行间距增大 50%。可以用如下命令来重定义该参数：

```
\renewcommand{\arraystretch}{因子}
```

### §4.8.3 表格样例

实际上表格的创建要比上面所列的复杂格式简单得多。有几个例子可以很好地演示这一点。

Position	Club	Games	W	T	L	Goals	Points
1	Amesville Rockets	33	19	13	1	66:31	51:15
2	Borden Comets	33	18	9	6	65:37	45:21
3	Clarkson Chargers	33	17	7	9	70:44	41:25
4	Daysdon Bombers	33	14	10	9	66:50	38:28
5	Edbartown Devils	33	16	6	11	63:53	38:28
6	Freeburg Fighters	33	15	7	11	64:47	37:29
7	Gadsby Tigers	33	15	7	11	52:37	37:29
8	Harrisville Hotshots	33	12	11	10	62:58	35:31
9	Idleton Shovers	33	13	9	11	49:51	35:31
10	Jamestown Hornets	33	11	11	11	48:47	33:33
11	Kingston Cowboys	33	13	6	14	54:45	32:34
12	Lonsdale Stompers	33	12	8	13	50:57	32:34
13	Marsdon Heroes	33	9	13	11	50:42	31:35
14	Norburg Flames	33	10	8	15	50:68	28:38
15	Ollison Champions	33	8	9	16	42:49	25:41
16	Petersville Lancers	33	6	8	19	31:77	20:46
17	Quincy Giants	33	7	5	21	40:89	19:47
18	Ralston Regulars	33	3	11	19	37:74	17:49

最简单的表格就是由行列组成的文本条目，其每一项要么居中，要么向一边对齐。列宽、列间距以及相应的表格宽度都是自动计算出来的。

上面这个表格有八列，第一列是右对齐，第二列是左对齐，第三列居中，接下来的三列又是右对齐，最后两列居中。因此在 `tabular` 环境中的列格式参数是

```
{rlcrrrcc}
```

生成这个表格的输入为：

```
\begin{tabular}{rlcrrrcc}
Position & Club & Games & W & T & L & Goals & Points\\[0.5ex]
1 & Amesville Rockets & 33 & 19 & 13 & 1 & 66:31 & 51:15 \\
2 & Borden Comets & 33 & 18 & 9 & 6 & 65:37 & 45:21 \\
... & ..... & .. & .. & .. & .. & ... & ... \\
17 & Quincy Giants & 33 & 7 & 5 & 21 & 40:89 & 19:47 \\
18 & Ralston Regulars & 33 & 3 & 11 & 19 & 37:74 & 17:49
\end{tabular}
```

在每一行中，各个列条目之间是用符号 `&` 分开的，行本身是用 `\\` 结束的。在第一行后面的 `[0.5ex]` 是增加前两行的竖直间距。最后一行不必用结束符号，因为当遇到 `\end{tabular}` 命令时会自动结束该行的。

在列格式参数中可以包含符号 `|`，以使得列之间用竖线分开。把第一行改为

```
\begin{tabular}{r|l||c|rrr|c|c}
```

那么结果为：

Position	Club	Games	W	T	L	Goals	Points
1	Amesville Rockets	33	19	13	1	66:31	51:15
2	Borden Comets	33	18	9	6	65:37	45:21
⋮	⋮						⋮
17	Quincy Giants	33	7	5	21	40:89	19:47
18	Ralston Regulars	33	3	11	19	37:74	17:49

在第一个列格式之前或者最后一个列格式之后的符号 `|` 都会在表格的外边生成一条竖线。两个符号 `||` 生成双竖线。可以用命令 `\hline` 生成与表格宽度相同的水平线。这一命令只能出现在一个行结束符 `\\` 后面或者第一行的开始。如此的两个命令 `\hline\hline` 会绘制双重水平线。

```
\begin{tabular}{r|l||c|rrr|c|c} \hline
Position & Club & Games & W & T & L & Goals & Points\\
\hline\hline
1 & Amesville Rockets & 33 & 19 & 13 & 1 & 66:31 & 51:15 \\
\hline
```

```

. . . . .
18 & Ralston Regulars      & 33 & 3 & 11 & 19 & 37:74 & 17:49 \\
\hline
\end{tabular}

```

那么现在的表格是：

Position	Club	Games	W	T	L	Goals	Points
1	Amesville Rockets	33	19	13	1	66:31	51:15
2	Borden Comets	33	18	9	6	65:37	45:21
:	:						:
17	Quincy Giants	33	7	5	21	40:89	19:47
18	Ralston Regulars	33	3	11	19	37:74	17:49

在这种情形中，由于在表格的最后有 `\hline`，因此必须给出行结束符 `\\`。

在这个例子中，所有行的第三列元素都是一样的，即 33。这样的公共条目可以利用列格式中的 `@{文本}` 形式的 `@-` 表达式自动插入，它把 文本 插入到相邻两列中。在我们的例子中，可以把列格式改成

```
{r|@{ 33 }rrrrcc} 或者 {r|l||@{ 33 }|rrr|c|c|}
```

这样文本 ‘33’ 就会连同空白一起出现在每行的第二列和第三列之间。这样得到了行条目稍有点儿不同的同样表格：例如，第四行现在的输入应为

```
4 & Daysdon Bombers      & 14 & 10 & 9 & 66:50 & 38:28 \\
```

列格式现在只是由 7 个列定义组成的：`rlrrrrcc`。原来对应第三列的 `c` 现在已去掉了，从而现在每行应少一个列分隔符 `&`。现在的第三列，即取胜的局数，是由第二个 `&` 开始的，而且其与俱乐部名称之间填充进 `@{ 33 }` 的内容，它不用额外的 `&` 符号就能自动插入进来。

后面两列以居中的形式给出了得失球与分数的关系。这里的冒号 ‘:’ 只是由于巧合上下位置一致，因为在所有行中冒号两边都是一个两位数。如果某一项为 `9:101`，那么由于该项要居中摆放，冒号就会向左稍微偏一点儿了。

也可以实现不依赖于数字位数，而仍然让 ‘:’ 上下对齐，这就要利用在列格式中 `r@{:}l` 形式的 `@-` 表达式。其意义为在每一行的一个右对齐和左对齐列之间放入冒号。那么例子中现在的列格式参数变为：

```
{r|@{ 33 }rrrr@{:}lr@{:}l}
```

或者

```
{r|l||@{ 33 }|rrr|r@{:}l|r@{:}l|}
```

而行条目变为：

```
4 & Daysdon Bombers      & 14 & 10 & 9 & 66 & 50 & 38 & 28 \\
```

先前的一个 `c` 列，现在被 `r@{:}l` 形式的两列所取代。`@-` 表达式就是在

相邻两列间插入文本，而且去掉通常情况下应该存在的列间距。因此 `r` 列是紧靠 ‘.’ 右对齐的，而 `l` 列是紧靠它左对齐的。

当一列是由具有小数点的长度不定的数字组成时，也可以用同样的方法达到小数点对齐的目的。

原来表示进球数和得分关系的条目现在由关于 ‘.’ 符号定位的两列组成。对于输入得失球或者得分数字时，这没有任何问题。然而，列标题是 ‘Goals’ 和 ‘Points’，其每个占两列的空间，中间没有冒号。可以用命令 `\multicolumn` 来解决这一问题，它把特定行中所选定的两列合并，并重新定义列格式。无方框的表格中第一行应该是：

```
Position & Club & W & T & L & \multicolumn{2}{c}{Goals}
& \multicolumn{2}{c}{Points} \\[0.5ex]
```

这里的 `\multicolumn{2}{c}{Goals}` 意味着接下来的两列被组合成一个居中的列，其中包含着文本 ‘Goals’。对于有框表格，在 `\multicolumn` 命令中新格式参数必须是 `{c|}`，因为当原来的列被合并时，竖线符号 `|` 也同时被去掉了。要想知道哪些东西属于给定的列，只要记住规则：一列拥有所有的直至下一个 `r`, `l` 或 `c` 为止的内容。

那么我们现在 1994/95 年橄榄球联赛表格有如下的标题：

1st Regional Soccer League — Final Results 1994/95							
	Club	W	T	L	Goals	Points	Remarks
1	Amesville Rockets	19	13	1	66:31	51:15	League Champs
2	Borden Comets	18	9	6	65:37	45:21	Trophy Winners
3	Clarkson Chargers	17	7	9	70:44	41:25	Candidates for National League
4	Daysdon Bombers	14	10	9	66:50	38:28	
5	Edbartown Devils	16	6	11	63:53	38:28	
6	Freeburg Fighters	15	7	11	64:47	37:29	Medium Teams
7	Gadsby Tigers	15	7	11	52:37	37:29	
8	Harrisville Hotshots	12	11	10	62:58	35:31	
9	Idleton Shovers	13	9	11	49:51	35:31	
10	Jamestown Hornets	11	11	11	48:47	33:33	
11	Kingston Cowboys	13	6	14	54:45	32:34	
12	Lonsdale Stompers	12	8	13	50:57	32:34	
13	Marsdon Heroes	9	13	11	50:42	31:35	
14	Norburg Flames	10	8	15	50:68	28:38	Disbanding
15	Ollison Champions	8	9	16	42:49	25:41	
16	Petersville Lancers	6	8	19	31:77	20:46	Demoted
17	Quincy Giants	7	5	21	40:89	19:47	
18	Ralston Regulars	3	11	19	37:74	17:49	

输入为：

```
\begin{tabular}{|r|l||rrr|r@{:}l|r@{:}l||c|}\hline
\multicolumn{10}{|c|}{\bfseries 1st Regional Soccer League ---
Final Results 1994/95} \\ \hline
```

```
&\itshape Club &\itshape W &\itshape T &\itshape L &
  \multicolumn{2}{c|}{\itshape Goals}
  & \multicolumn{2}{c||}{\itshape Points}
  & \itshape Remarks \\ \hline\hline
```

从 3-5，7-14 和 17 位的水平线是用命令 `\cline{1-9}` 生成的，而其它地方的则是用 `\hline` 生成的：

```
11 & Kingston Cowboys      & 13 & 6 & 14 & 54&45 & 32&34 &
    Medium Teams \\ \cline{1-9}
```

对最后两行需要特别说明一下。备注 ‘Demoted’ 在竖直方向恰好位于两行的中间。这是用如下输入来得到的：

```
18 & Ralston Regulars      & 3 & 11 & 19 & 37&74 & 17&49
    & \raisebox{2.3ex}[0pt]{Demoted}\\ \hline
```

`\raisebox` 命令把文本 ‘Demoted’ 向上提升 `2.3ex`。这里如果没有可省参数 `[0pt]`。那么这里对盒子的提升会导致最后一行的总高度也增加了 `1.5ex`。这就会使得在第 17 行下面的水平线与第 18 行文本之间的竖直距离增大。我们就是用可省参数 `height = [0pt]` 来抑制这一额外间距。（参见 4.7.2 节中关于 `\raisebox` 命令的详细描述。）

有时我们需要增大水平直线与包围文本之间的竖直距离。如果前面那个表格的标题为如下形式时，就会更好看些：

1st Regional Soccer League — Final Results 1994/95							
	<i>Club</i>	<i>W</i>	<i>T</i>	<i>L</i>	<i>Goals</i>	<i>Points</i>	<i>Remarks</i>

这可以通过在标题文本中插入一个不可见的竖直标尺，即支撑（4.7.6 节）来做到这一点：

```
\multicolumn{10}{l|}{\rule[-3mm]{0mm}{8mm}\bfseries 1st
  Regional Soccer League --- Final Results 1994/95} \\ \hline
```

这个插入进来的标尺宽度为 0 mm，所以它是不可见的，它向基线下面伸展了 3 mm，有 8mm 高。因此它向基线上面伸展了 5mm（8 - 3）。因此它非常有效地把两个方向的水平线推离了标题。如果一行中不只一列，那么只要在一列中包含支撑就可以了，因为整行的尺寸是由最大列决定的。

**练习 4.14:** 用与上面排版橄榄球比赛结果一样的方式，生成一个你自己钟爱的比赛项目结果表。注意确保正确对齐得失球与得分中 ‘:’ 的关系。

**练习 4.15:** 生成下面这个时间表，其中条目 ‘Day’ 和 ‘Subj.’ 等提升的高度与前面表格中 ‘Demoted’ 的一样。为了简化使用，可以引进如下用户定义的命令：

`\newcommand{\rb}[1]{\rasiebox{2.3ex}[Opt]{#1}}`

(见 7.3.2 节), 这样 `\rb{条目}` 的作用就和 `{\rasiebox{1.5ex}[Opt]{条目}}` 一样。这条命令也同样可以用于其它需要提升的条目, 如 `\rb{ Mon. }` 或者 `\rb{UNIX}` 等等。

Day	6.15–7.15pm		7.20–8.20pm		8.30–9.30pm	
	Subj.	Teacher	Subj.	Teacher	Subj.	Teacher
		Room		Room		Room
Mon.	UNIX	Dr. Smith	Fortran	Ms. Clarke	Math.	Mr. Mills
		Comp. Ctr		Hall A		Hall A
Tues.	I <sup>A</sup> T <sub>E</sub> X	Miss Baker	Fortran	Ms. Clarke	Math.	Mr. Mills
		Conf. Room		Conf. Room		Hall A
Wed.	UNIX	Dr. Smith	C	Dr. Jones	ComSci.	Dr. Jones
		Comp. Ctr		Hall B		Hall B
Fri.	I <sup>A</sup> T <sub>E</sub> X	Miss Baker	C++	Ms. Clarke	canceled	
		Conf. Room		Conf. Room		

在上面所有的例子中, 在每个列中的条目都只有一行。而有的表格中某些列包含多行文本。

Model	Description	Price
GXT 1	<b>The PC Compatible:</b> Intel 80386, 1 MB main memory, color graphic card, multi-I/O card, 2 drives 1.2 MB, 17" monitor, keyboard, MS-DOS 6.2, GW Basis	883.70
GXT 20	<b>The XT Compatible:</b> Data as for GXT, but with Hercules compatible card, 1 drive 1.2 MByte, 1 hard disk 120 MByte unformatted	1376.40
GAT	<b>The AT Compatible:</b> Intel 80486, 1 drive 1.2 MB/720 KB, 8 MByte main memory, 17" monitor, keyboard, Hercules compatible card, hard disk 240 MByte unformatted, Serial/Parallel card, MS-DOS 6.2, GW Basic	2356.00

上面这个表格由三列组成, 第一列左对齐, 第三列右对齐。中间的那列包含着多行文本, 每行宽度为 8.0cm。这是用列格式符号 `p{ 宽度}` 来做到的。在这个表格中全部的列格式参数是 `{lp{8.0cm}r}`。

```
\begin{tabular}{lp{8.0cm}r}
\bfseries Model & Description & \bfseries Price \\[1ex]
GXT 1 & \small{\bfseries The PC Compatible}: Intel
80386, 1 MB main memory, color graphic card,
multi--I/O card, 2 drives 1.2 MB, 17'' monitor,
keyboard, MS-DOS~6.2, GW Basis & 883.70\\
```



```

. . . . .
MS-DOS~6.2, GW Basic & 2356.00
\end{tabular}

```

中间那列的文本只需简单输入就可以了，自动被断成 8.0cm 宽的行。该列就像通常那样用 & 符号与其它列分开。

警告：在 p 列中不能用行结束符 \\，因为它会被解释为表格行的结束。而可以用断行命令 \newline 和 \linebreak。然而，如果某一行确实需要用 \\ 来断开，那么整个列就要放在 \parbox 中，其宽度与 p 列的一样。列条目可以由几段组成，空行表示分段。

练习 4.16: 生成下面这个表格。

Course and Date	Brief Description	Prerequisites
Introduction to LSEDIT March 14–16	Logging on — explanation of the VMS file system — explanation and intensive application of the VMS editor LSEDIT — user modifications	none
Introduction to L <sup>A</sup> T <sub>E</sub> X March 21–25	Word processors and formatting program — text and commands — environments — document and page styles — displayed text — math equations — simple user-defined structures	LSEDIT

最后这个例子描述了一个有框表格的空白表。这儿的困难在于设置空白盒子的高度和宽度，因为通常这些量是由文本条目自动确定的。这个例子说明了如何借助于支撑和 \hspace 命令来做到这一点。

Budget Plan 1995–1997																							
Project	Nr. <table><tr><td></td><td></td><td></td></tr></table>						Name <table><tr><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr></table>																
Year	1995			1996			1997																
	(GB £)	US \$		(GB £)	US \$		(GB £)	US \$															
Investment																							
Costs																							
Operating																							
Costs																							
Industrial																							
Contract																							
Signature						Authorization																	

```

\newsavebox{\kk}\newsavebox{\kkk}
\sbox{\kk}{\framebox[4mm]{\rule{0mm}{3mm}}}
\sbox{\kkk}{\usebox{\kk}\usebox{\kk}\usebox{\kk}}
\begin{tabular} { | l | c | c | c | } \hline

```

```

\multicolumn{4}{|c|}{\rule[-0.3cm]{0mm}{0.8cm}\bfseries
    Budget Plan 1995--1997}\
\hline\hline
\rule[-0.4cm]{0mm}{1cm}Project
& \multicolumn{3}{|l|}{Nr. \usebox{\kkk}\hspace{0.5cm}}
\vline\hspace{0.5cm}Name\usebox{\kkk}\usebox{\kkk}\usebox{\kkk}
\usebox{\kkk}}\ \hline
\multicolumn{1}{|r|}{Year} & 1995 & 1996 & 1997 \\\
\cline{2-4}
& (GB \pounds) \vline\ US \$ & (GB \pounds) \vline\ US \$
& (GB \pounds) \vline\ US \$ \\\hline
Investment & \hspace{2.5cm} & \hspace{2.5cm} & \hspace{2.5cm} \\\
Costs & & & \\\hline
Operating & & & \\\
Costs & & & \\\hline
Industrial& & & \\\
Contract & & & \\\hline
\multicolumn{4}{|l|}{\rule[-1.2cm]{0mm}{1.5cm}Signature
    \hspace{5cm}\vline~Authorization} \\\hline
\end{tabular}

```

前三行只是直接与表格的结构有关。当使用了命令 `\usebox{\kkk}` 时就会画出三个空白框（见 4.7.1 节）。

我们用 `\hspace{2.5cm}` 命令来设置后三列的宽度，用 `\vline` 在一列中画一条竖直线，除这两点外，这个例子同前面的例子相比，再没有什么新的地方。这里只需要对最后一行加以解释：

命令 `\multicolumn{4}{|l|}` 把全部四列合并为一列，文本被设置成左对齐。其文本中首先是一个支撑 `\rule[-12mm]{0mm}{15mm}`，这也就是说最后这行的高度开始于基线下面 12mm，总高度为 15mm。接着，从左页边开始，显示单词 `Signature`，间隔 5cm 后是 `\vline`，即一条竖线。单词 `Authorization` 与竖线之间用一个空白（~）隔开。

上面的例子清楚地说明了表格中列宽度和行高度是如何自动确定的。然而这些尺寸也会受支撑和 `\hspace` 命令的影响。而且在 4.8.2 节中描述的命令除了可以改变线粗细外，还可以增加列间距和行间距。例如，

```
\setlength{\tabcolsep}{5mm}
```

就会在每列的前后插入 5mm 的间距；即得到了 10mm 的列间距。4.8.2 节中有关于如何使用这些表格样式参数的说明。

## §4.8.4 浮动表格

`tabular` 环境是在其所出现的地方生成一个表格，紧接其前面的文本，后面就是跟着它的其它内容。当在页面上表格与周围文本匹配得很好时，这没有什么问题，而且这通常也说是我们想要的。然而，如果表格很长，从它定义的地方开始，在一页中是放不下来的，那么就会结束该页，下一页开头就是这个表格，后面再接后续文本。这样就会导致当前页的格式并不是希望看到的样子。

如果在当前页上表格出现的地方有足够空间放下这个表格，那么就把它马上放下，否则就继续排版文本，保留表格到当有足够空间时，如下一页的开头，再排版表格，这种方式就是很好的。因于表格通常有标题和 / 或题目，这些项也自然应该随着它一起移动。

L<sup>A</sup>T<sub>E</sub>X 提供了浮动表格（还有插图）的功能，其附加文本也以同样方式移动。这可以用如下环境来实现此功能：

```
\begin{table} 上部文本 表格 下方文本 \end{table}
```

这里的 `表格` 代表诸如 `tabular` 环境定义的整个表格，`上部文本` 表示出现在表格上方的文本，而 `下方文本` 表示出现在表格下方的文本。与表格相关联文本的宽度、间距和位置都要由用户来安排。

出现在 `\begin{table}` 和 `\end{table}` 中的所有内容是包围它的外部文本无关，它们通常都放在当前页的开头。如果已有一个表格占用了这个位置，那就会尝试是否可能有足够的空间，把它放在当前页的底部。否则，就会把它放在下一页上，这样就有可能积累很多表格。包围表格的文本就如同没有表格那样进行排版。关于浮动的一般性细节，以及自动有序编号，请见 6.6 节。

在本页顶部的表格就是由放在此处的浮动表格生成的，其输入文本为（这里没有列出脚注的生成，我们将在 4.10.4 节中介绍）：

```
\begin{table}{\bfseries Primary Energy Consumption}\\[1ex]
  \begin{tabular*}{118mm}{@{}11...rr@{}}
    . . . . .
  \end{tabular*}\\[0.5ex]
  \emph{Source:} Energy Balance Study Group, . . .
\end{table}
```

有许多格式参数可以与 `table` 环境给合使用，我们将把它们与插图相应的参数一起在 6.6 节中介绍。

**练习 4.17:** 补全上面的生成浮动表格的文本（不要脚注）。注意下面的问题（必要时可查看在 4.8.1 节中关于 `@-` 表达式的解释）：

1. 在格式定义中开头和结尾的 `@{}` 有什么作用？

**Primary Energy Consumption**

Energy Source	1975	1980	1986
Total Consumption			
(in million tons of BCU <sup>a</sup> )	347.7	390.2	385.0
of which (percentages)			
petroleum	52.1	47.6	43.2
bituminous coal	19.1	19.8	20.0
brown coal	9.9	10.0	8.6
natural gas	14.2	16.5	15.1
nuclear energy	2.0	3.7	10.1
other <sup>b</sup>	2.7	2.3	3.0

<sup>a</sup>BCU- Bituminous Coal Unit(1 ton BCU corresponds to the heating equivalent of 1 ton of bituminous coal = 8140 kwh

<sup>b</sup>Wind, water, solar energy, etc.

Source: Energy Balance Study Group, Essen 1987.

2. `tabular*` 环境生成一个给定宽度的表格，这里是 118mm。在格式定义开头处的 `\@{\extracolsep{\fill}}` 的作用是什么呢？
3. 为了得到这里所显示的表格样式，`\@{\extracolsep{\fill}}` 和相反的命令 `\@{\hspace{1em}}\@{\extracolsep{1em}}` 应该出现在格式定义的什么地方呢？如果只有 `\@{\extracolsep{1em}}` 做为相反命令，那么表格会是什么样子呢？

**§4.9 显示源文本**

有时我们会希望某些文本直接按输入的样子显示，即不进行格式化处理。这可以用如下命令来实现：

```
\begin{verbatim}      源文本      \end{verbatim}
\begin{verbatim}*      源文本      \end{verbatim}
```

这样 源文本 就会同输入时的一样，用打字机字样显示出来，同时包含所有的空格和断行。通常情况下被当做命令的特殊字符也做为字符串直接显示出来。唯一的例外就是标志着该环境结束的命令 `\end{verbatim}` 本身。这里的显示是在新的一行上开始，然后再在新一行上继续 `verbatim` 环境后的正常文本。

这个环境的标准形式与 `*` 形式的差别在于，对后一种情形，空格是用符号 `␣` 表示的，以使得空格更容易看到。

例如，在 97 页上就有一些源文本被直接显示出来，以演示在禁止模式中脚注的应用。这可以用如下方法得到：

```
\begin{verbatim}
  \addtocounter{footnote}{-1}\footnotetext{Small insect}
  \stepcounter{footnote}\footnotetext{Large mammals}
\end{verbatim}
```

源文本也可以用如下命令在一行里显示出来：

```
\verb<源文本 c
```

```
\verb*<源文本 c
```

这里的 *c* 为不出现于 源文本 中的任一字符。它是用来终止这种原样显示模式，返回如通常那样进行正常的文本格式。在 `\verb` 或者 `\verb*` 命令与 *c* 之间不能用空格，显然，对于普通形式，*c* 不能是 `*`。

要显示普通文本中的命令名称，可以用下面这种方法。例如，为了得到 `'\xyz{ }'`，输入应该是 `\verb=\xyz{ }=`。这里的等号 `=` 就用来做为定界符 *c*。`*`-形式的作用类似，只是同 `verbatim*` 环境一样，要把空格显示为 `\`：  
`\xyz{\ }`。

标准的 L<sup>A</sup>T<sub>E</sub>X 2<sub>ε</sub> 宏包 `shorvrb` (第 217 页) 提供了 `\verb` 命令的一种简凑而方便的版本。

同 `verbatim` 环境相比，在这里的 源文本 中不能有断行。如果其中的文本长度超过一行，断行点就会被当做空白对待 (L<sup>A</sup>T<sub>E</sub>X 2.09) 或者给出一个错误信息 (L<sup>A</sup>T<sub>E</sub>X 2<sub>ε</sub>)。这个错误信息就会提示你可能忘记用结束的 *c* 符号了。

重要：无论是 `verbatim` 环境，还是 `\verb` 命令都不能用作其它任何命令的参数。

练习 4.18: 从本书中按源文本复制一些文本行。

## §4.10 脚注和边注

### §4.10.1 标准脚注

脚注是用如下命令生成的：

```
\footnote{脚注文本}
```

该命令紧接在需要在脚注中进行解释的单词后面。脚注文本 用较小的字样以脚注的形式显示在当前页的底部。脚注的第一行要进行缩进，并给出与正文插入点处相同的脚注标记。在一页上的第一个脚注是用一条短水平线与正文分开的。

标准脚注标记是一个小的偏上的数字<sup>1</sup>，它是顺序编号的。

---

<sup>1</sup> 在一个用打字机打印的草稿中，通常是用 `*`, `**` 等等来标记脚注，这里也可以使用这种方法；然而，由于当输入文本时并不知道将在哪儿分页，因此这里就存在着如何避免在一页上有相同标记的问题。

... 小的偏上的数字 `\footnote{` 在一个用打字机打印的草稿中，通常是用... 有相同标记的问题。}，它是顺...

在 `article` 类中，整个文档统一对脚注进行编号，而在 `report` 和 `book` 类中，每当开始新的一章时其都会重置为 1。

`\footnote` 脚注只可以位于通常的段落模式中，不能位于数学模式或 LR 模式（1.5.3 节）中。实际上，这就意味着它不能位于一个 LR 盒子（4.7.1 节）或者子段盒子（4.7.3 节）中。然而，它却可以用在 `minipage` 环境中，此时脚注文本显示在小页的下面，而不是实际页面的底部。<sup>2</sup>

`\footnote` 命令必须紧接在接受这个注释的单词后面，中间不必有任何空格或间隔。一句话的脚注可以在句号后面给出。

### §4.10.2 非标准脚注

如果用户希望在 `article` 类中每当开始新的一节时，脚注编号重置为 1，那可以用如下命令：

```
\setcounter{footnote}{0}
```

就把这条命令放在 `\section` 命令的前面或后面。

内部脚注记数器的名称为 `footnote`。每次调用 `\footnote` 都会使这个记数值增 1，并以阿拉伯数字形式显示出新值做为脚注的标记。可以用如下命令来实现不同样式的标记：

```
\renewcommand{\thefootnote}{\数字样式{footnote}}
```

这里的数字样式就是一个在 4.3.5 节中所描述的记数器显示命令：`\arabic`，`\roman`，`\Roman`，`\alph` 或 `\Alph`。然而，对于 `footnote` 记数器，还有另一个记数器显示命令可以使用，它就是 `\fnsymbol`，它把从 1 到 9 的记数器值显示为如下 9 个符号：

```
* † ‡ § ¶ || ** †† ‡‡
```

这里需要用户在调用第十个 `\footnote` 命令之前把脚注记数器重值为零。

在 `\footnote` 命令中还可以有一个可省参数：

```
\footnote[数]{脚注文本}
```

这里的数是一个正数，要用它来取代显示标记的脚注记数器的值。在这种情况下，脚注记数器的值并没有增加。例如 \*\*，

```
\renewcommand{\thefootnote}{\fnsymbol{footnote}}
```

例如 `\footnote[7]{第七个标记符号。}`，

```
\renewcommand{\thefootnote}{\arabic{footnote}}
```

<sup>2</sup>对于嵌套的小页，脚注会在遇到第一个 `\end{minipage}` 命令时显示出来，这样会导致出现在错误的地方。

\*\*第七个标记符号。

这里的最后一行是为了把脚注标记样式恢复成标准形式。否则，后面所有的脚注都以符号为标记，而不是用数字。

### §4.10.3 禁止模式中的脚注

可以用如下模式在文本中插入一个脚注标记：

```
\footnotemark[ 数 ]
```

即使这里的文本中不允许脚注，例如 LR 盒子，表格和数学模式中。这里的标记就是相应于可以省略的参数 数 或者当这个参数省略时脚注记计数器增加后的值。脚注本身并没有生成。它必须在禁止模式外面用如下命令生成：

```
\footnotetext[ 数 ]{ 脚注文本 }
```

如果在脚注标记中已用了可省参数，那么在文本命令中也必须用相同的 数 做选项。类似地，如果在标记中没有用选项，在文本中也不能出现选项。这儿的脚注生成时将利用 数 的值或者脚注记计数器的值。

当 \footnotemark 命令没有可省参数时，调用一次会使记计数器增 1 。而相应的 \footnotetext 命令并不改变记计数器的值。

如果在接下来的 \footnotetext 命令前面有许多 \footnotemark 命令，其都没有带可省参数，那么就需要用如下命令调整记计数器的值：

```
\addtocounter{footnote}{ 差 }
```

这里的 差 是一个负数，它表示记计数器必须被减少多少。因此在每次调用命令 \footnotetext 之前，记计数器必须增 1 。这就可以用 差= 1 的 \addtocounter 命令或者

```
\stepcounter{footnote}
```

命令来给记计数器增 1 。

For example: mosquitoes<sup>3</sup> and elephants<sup>4</sup>

```
For example: \fbox{mosquitoes\footnotemark\ and
elephants\footnotemark}
```

生成脚注标记 <sup>3</sup> 和 <sup>4</sup> 。那么现在记计数器的值为 4 。为了使有框盒子外面的第一个 \footnotetext 相应于正确的记计数器值，现在要把它减 1 。这样两个脚注文本就可以如下生成：

```
\addtocounter{footnote}{-1} \footnotetext{Small insects}
```

```
\stepcounter{footnote}\footnotetext{Large mammals}
```

把它们就放在 \fbox{} 命令的后面。现在脚注记计数器的值与它离开 \fbox 时的相同。

---

<sup>3</sup>Small insects

<sup>4</sup>Large mammals

#### §4.10.4 小页环境中的脚注

在 4.10.1 节中已提到，在小页环境中可以用脚注命令。然而脚注是显示在小页的下面，而不是当前页的底部。

在小页环境 <sup>a</sup> 中的脚注命令有不同的标记样式。脚注会在接下来第一个的

`\end{minipage}` 命令时显示出来。<sup>b</sup>

小页环境中的脚注与正文所用的脚注使用不同的记数器，它的记数器称为 `mpfootnote`，它的记数与 `footnote` 无关。

`\begin{minipage}{6cm}`

在小页环境 `\footnote{` 标记是偏上的小写字母。`}`

中的脚注命令有不同的...

<sup>a</sup>标记是偏上的小写字母。

<sup>b</sup>当小页环境嵌套时这有可能导致麻烦。

在表格即 `tabular` 环境中的脚注通常只能用上面所描述的命令得到：

`\footnotemark` 在表格内部，`\footnotetext` 在环境外面。然而如果 `tabular` 环境是在一个小页环境内部，通常的 `\footnote` 命令也可以在表格内部使用。脚注显示在小页结束时的表格下方。

练习 4.19: 在标准练习文件中的适当地方插入一些脚注，并选择适当的脚注文本。

练习 4.20: 重定义命令 `\thefootnote`，使得脚注标记变为在 4.10.2 节中所描述的符号。把这个定义加到标准练习文件的导言中。

练习 4.21: 完成练习 4.17，使得在 93 页上的表格中有脚注 <sup>a</sup> 和 <sup>b</sup>。

#### §4.10.5 脚注样式参数

有两个脚注样式参数可以在需要时进行修改，这种修改既可以在导言中进行，也可以在一个环境中进行。

`\footnotesep`

两个脚注间的竖直距离。可以用 `\setlength` 命令修改这个长度。

`\footnoterule`

这条命令在正文与脚注之间画一条水平线。它并不会增加任何竖直间距。可以修改它的定义，例如，

`\renewcommand{\footnoterule}`

`{\rule{宽度}{高度}\vspace{-高度}}`

当 `高度` 的值为零时就会生成一个零高度的不可见直线。

#### §4.10.6 边注

在页边上的注释可以用下面的命令生成：

`\marginpar{注释文本}`



它把 注释文本 放在右面的页边上, 开始点与命令所在行相平。这里所出现的边注 就是用如下输入得到的:

... 这里所出现的边注 `\marginpar{ 这是 \\ 一个 \\ 边注 }` 就是 ... 一个

这里的 注释文本 通常放在一个宽度为 1.9cm(0.75in) 的子段盒子中。这样窄的盒子通常会使得断行非常困难, 这也就是上面为什么要用 `\\` 命令人为断行的缘故。因此在边注中, 这样的盒子对于只有一个符号就是非常合适的, 如这里所显示的箭头。

另一种使用需要使用边注的地方就是在要引起注意的文本旁边画一条竖线。这有时用来表示相对于以前的版本, 这部分文本进行了改动。本段中的标记样例就是在第一行中用如下输入得到的:

```
\begin{marginpar}{\rule[-17.5mm]{1mm}{20mm}}
```

边注的宽度可以用下一节所描述的样式参数进行改动。用户必须保证总宽度不能变得超过打印机所接受的限度。

在默认状态下, 边注出现在一页的右边, 或者当选定了 `twoside` 选项时的外页边。这里的‘外’指的是奇数页的右页边, 偶数页的左页边。当用了 `twocolumn` 选项时, 它们就被放在外面的边界上: 左列的左边, 右列的右边。

但这样做有时会使类似上面给出的箭头这样的页边记号出现问题。如果该页为偶数页, 它就必须指向相反的方向。事实上, 它的方向是与它位于页的哪一边有关, 也就是依赖于页码和列。由于在书写 (或者后来对它进行修改) 时并不知道它指向它哪个方向。这可以用 `\marginpar` 命令的扩充语法来实现这一点:

```
\marginpar[ 左文本 ]{ 右文本 }
```

这种形式的命令包含页边文本的两个版本, 左文本 会出现在左边, 右文本 会在出现在右边, 具体视需要哪个而定。因此为了全面起见, 排版上面那个箭头的边注命令应该是

```
\marginpar[\hfill$\Longrightarrow$]{ $\Longleftarrow$ }
```

(箭头命令是数学符号, 在 5.3.5 节对它们进行详细介绍。)

在上面的 `\marginpar` 例子中若没有用 `\hfill` 命令, 那么位于左边界上的箭头就会显示在其所在段的左边, 从而离正文很远。之所以会这样, 是因为 `\marginpar` 命令把它的内容左对齐处理, 在下一页上的一个有框边注就很好地说明了这一点: 靠左边对齐正文的方式只适用于边注位于右边界的情形; 如果它位于左边界, 就会离正文太远。`\hfill` 命令就是为了使得边注内容向右边对齐, 从而使得它处在相对于正文比较恰当的位置上。

对于这一节第一个边注, 也需要用同样的方法来处理。实际生成它的命令是

```
\marginpar[\flushright 这是 \\ 一个 \\ 边注 ]{ 这是 \\ 一个 \\ 边注 }
```

这里的 `\flushright` (4.2.2 节) 就等价于在每一行上放一个 `\hfill`。

边注的标准位置可以用 `\reversemarginpar` 命令来进行变换。一旦使用了这个命令, 边注就会位于左边, 或者在 `\twoside` 选项时出现在内边界。

⇒ 这个命令的作用直到出现相反命令 `\normalmarginpar` 为止。对于 `twocolumn` 命令, 这两个命令没有作用。

在边注中不能进行分页。如果一个边注太靠近页面底部, 从而没有足够空间放下它, 那边注就会延伸到最后一行的下面。在这种情形中, `\marginpar` 命令中的文本开头应包含一个 `\vspace` 命令, 以使得边注向上移动一点儿, 或者就用两条 `\marginpar` 命令把它分成两部分, 分放在不同的页上。这种手工调整只有在整篇文档完成后才能进行, 因为以后对文档进行改动就会使情形发生变化。

#### §4.10.7 边注的样式参数

可以改动下面的样式参数, 以重定义边注的显示方式:

`\marginparwidth`

定义边注盒子的宽度;

`\marginparsep`

设置边注盒子与正文边界之间的距离;

`\marginparpush`

两个边注盒子之间的最小竖直距离。

这些参数都是长度, 可以用 `\setlength` 命令像通常那样赋予一个新值。

### §4.11 正文中的注释

在正文中若有注释、解释、提示等等的东西, 对于告诉作者或者其它后来由于某一目的进行某种构造的用户而言, 是非常有用的。这些注释不与其它的正文一样被格式化。

在  $\text{T}_{\text{E}}\text{X}$  中, 单个字符命令 `%` 就引进了注释。当这个字符出现在正文中, 它自己本身以及该行后面的其它文本都要被忽略。如果一个注释有几行长, 那么每行都必须前缀 `%`。

注释符号 `%` 对暂时抑制某些命令也是很有用的。把一个 `%` 放在这样一条命令的前面, 那么就会使该行后面部分都被忽略。这称为 *注释抑制行*。类似地, 在源文件中可能有不同的替换文本, 因此可以注释掉一些行, 这样作者可以决定要用哪一版本。

最后提一下, `%` 符号对去掉每行结尾的那个隐含空格有相当重要的作用。在用户定义中当希望在两个参数间不要由于分行而引进不必要的空格时, 它是特别有用的。

**练习 4.22:** 注释掉在练习 4.20 中对导言做的改动。以后可以通过去掉 % 符号来重新激活这些命令。