第 6 章　LaTeX 的私人订制

Du Ang
du2ang233@gmail.com

2017 年 8 月 6 日

目录

之前学到的一些命令看似都够用了，但是它们的输出的效果有时并不是特别好看。而且有的时候，LaTeX 提供的命令或环境并不能满足我们的要求。试想：我要如何制作一个简单但像样的毕业论文/书籍/简历模板，每次都可以直接套用，而不是再在导言区写一堆代码?

这一章的内容可以帮我们实现这一目标，让我们编写可重复利用的模块——宏包和文档类，并在其中自己定义命令和环境，让 LaTeX 产生不同于默认的输出。

1 自定义命令、环境、宏包和文档类

1.1 定义新命令

定义新命令：

```
\newcommand{name}[num]{definition}
```

该命令需要两个基本的参数：新命令的名字 *name* 和新命令的定义 *definition*。可选参数 *num* 可以指定新命令的参数数目（最多 9 个）。如果不写 *num* 参数，默认为 0，表示新命令没有参数。

示例 1 定义了一个新命令 `\tnss`，是“The Not So Short Introduction to LaTeX 2_ε”的缩写。

示例代码 1：

```
\newcommand{\tnss}{The not so Short Introduction to \LaTeXe}
% in the document body:
This is "\tnss" \ldots{} "\tnss"
```

示例输出 1：

This is ”The not so Short Introduction to LaTeX 2_ε” … ”The not so Short Introduction to LaTeX 2_ε”

示例 2 定义了包含参数的命令。使用定义的命令的时候，在 `#1` 的位置指定参数。如果需要定义更多的参数，使用 `#2`、`#3`，以此类推。

示例代码 2：

```
\newcommand{\txsit}[2]{This is the \emph{#1} #2 Introduction to \LaTeXe}
% in the document body:
\begin{itemize}
    \item \txsit{not so}{short}
    \item \txsit{very}{long}
\end{itemize}
```

示例输出 2：

- This is the *not so* short Introduction to LaTeX 2_ε
- This is the *very* long Introduction to LaTeX 2_ε

LaTeX 不允许定义一个与现有命令重名的命令。如果要修改命令定义，使用 `\renewcommand` 命令。它的用法与 `\newcommand` 相同。

在某些情况下，可能会用到 `\providecommand` 命令。在命令不存在时，它相当于 `\newcommand`；在命令已经存在时，仍沿用原来已存在的定义。

1.2 定义新环境

和 \newcommand 命令类似，可以通过 \newenvironment 命令来定义环境：

```
\newenvironment{name}[num]{before}{after}
```

同样地，\newenvironment 有一个可选的参数 *num*。*before* 参数中的内容将在此环境包含的文本之前处理，*after* 参数中的内容将在遇到 \end{name} 命令时处理。

示例代码：

```
\newenvironment{king}
    {\rule{1ex}{1ex} \hspace{\stretch{1}}}
    {\hspace{\stretch{1}} \rule{1ex}{1ex}}
% in the document body:
\begin{king}
    My humble subjects \ldots
\end{king}
```

示例输出：

■ My humble subjects … ■

num 参数的用法和 \newcommand 命令相同。同样，也不能定义一个现有的环境，如果想修改某个现有的环境，使用 \renewenvironment 命令，语法和 \newenvironment 相同。

上面示例中的 \rule、\stretch 和 \hspace 命令后面会介绍。

1.3 多余的空格

在定义一个新环境的时候，很容易会有多余的空格“悄悄溜进来”。这样的小瑕疵有些时候却很致命，例如我们想创建一个没有缩进的标题环境、并且希望紧跟这个环境后面的段落也不要缩进的时候。\ignorespaces 命令会忽略环境开始部分的所有空格。想在环境结束后不留空格有点麻烦，可以用 \ignorespacesafte 命令，它的作用是在环境的‘end’之后执行 \ignorespaces。示例代码：

```
\newenvironment{simple}
    {\noindent}
    {\par\noindent} % \par ends the paragraph
% in the document body:
\begin{simple}
    See the space \\ to the left.
\end{simple}
Same \\ here.

\newenvironment{correct}
    {\noindent\ignorespaces}
    {\par\noindent\ignorespacesafterend}
% in the document body:
```

```
\begin{correct}
    No space \\ to the left
\end{correct}
Same \\ here.
```

示例输出：

See the space
to the left.

Same
here.

No space
to the left

Same
here.

1.4 LaTeX 的命令行参数

如果用的是类 Unix 操作系统，就可能会用 Makefile 来构建 LaTeX 项目，然后就可以用不同的命令行参数使使同一份文档编译出不同的版本。如果在文档中加入下面的代码：

```
\usepackage{ifthen}
\ifthenelse{\equal{\blackandwhite}{true}}{
    % "black and white" mode; do something...
}{
    % "color" mode; do something different...
}
```

然后利用下面的命令调用 LaTeX：

```
latex '\newcommand{blackandwhite}{true}\input{test.tex}'
```

首先 `\blackandwhite` 命令会被定义，然后真实的文件会被读取。如果将 `\blackandwhite`设置为 false，文档就会输出彩色版本。

1.5 定义新宏包

如果定义了很多的新环境和新命令，文档的导言区就会很长。这时，就可以定义一个 LaTeX 宏包来包含所有新环境和新命令的定义。需要时，再在文档中通过 `\usepackage` 命令调用。

如果想要自定义一个宏包，基本的工作是要将原来写在文档导言区的很长的内容拷贝到一个 `.sty` 文件中，并且需要在文件最前面加上 `\ProviedesPackage{package name}` 命令，这个命令可以在多次包含宏包的问题提示错误。其中，*package name* 要和我们定义的宏包名相同。

示例代码：

```
% Demo Package by Tobias Oetiker
\ProvidesPackage{demopack}
\newcommand{\tnss}{The not so Short Introduction to \LaTeXe}
\newcommand{\txsit}[1]{The \emph{#1} Short Introduction to \LaTeXe}
\newenvironment{king}{\begin{quote}}{\end{quote}}
```

如果想进一步把各种宏包的功能汇总到一个文件里，而不是在文档的导言区罗列一大堆宏包的话，LaTeX 允许我们在自己编写的宏包中调用其他宏包，命令为 `\RequirePackage`，用法和 `\usepackage` 一致：

```
\RequirePackage[options]{package name}
```

1.6 定义新的文档类

再进一步，如果想编写自己的文档类，如论文模板等，问题就稍稍麻烦一些了。首先，要把自定义的文档类文件以 `.cls` 作为扩展名，开头使用 `\ProvidesClass` 命令：

```
\ProvidesClass{class name}
```

同样地，*class name* 也要和文档类的文件名一致。

但是有了上述命令和之前学到的一些命令，还不足以完成一个文档类的编写。因为诸如 `\chapter`、`\section` 等等许多命令都是在文档类中定义的。事实上，许多时候我们只需要像调用宏包那样调用一个基本的文档类，这样可以省去许多不必要的麻烦。在文档类中使用其他文档类的命令是 `\LoadClass`，用法和 `\documentclass` 十分相像：

```
\LoadClass[options]{package name}
```

2 字体和字号

2.1 改变字体的命令

LaTeX 默认会根据文档的逻辑结构（章节、脚注等）自动选择适当的字体和字号。但是如果我们想自己手动地改变它们，这时就可以用表 1 和表 2。每种字体的真实大小是由文档类和它的设置决定的。表 3 展示了这些命令的绝对磅值，是由标准文档类定义的。

示例代码：

```
{\small The smal and \textbf{bold} Romans ruled}
{\Large all of great big \textit{Italy}.}
```

示例输出：

The smal and **bold** Romans ruled all of great big *Italy*.

LaTeX 2_ε的一个重要特点是字体的属性是独立的。意思是，如果一段文字事先被设置了粗体或斜体，后来又改变了它的字号或者字体，它会在原来粗体或斜体设置的基础上变化。

在数学模式中，使用改变字体命令可以暂时地退出数学模式，并且输入一些正常的字体。如果想换成另一种数学排版字体，需要使用到表 4 中的一些命令。

表 1: 字体命令

\textrm{...}	roman	\textsf{...}	sans serif
\texttt{...}	typewriter		
\textmd{...}	medium	\textbf{...}	**bold face**
\textup{...}	upright	\textit{...}	*italic*
\textsl{...}	*slanted*	\textsc{...}	Small Caps
\emph{...}	*emphasized*	\textnormal{...}	document font

表 2: 字号命令

\tiny	tiny font	\Large	larger font
\scriptsize	very small font	\LARGE	very large font
\footnotesize	quite small font	\huge	huge
\small	small font	\Huge	largest
\normalsize	normal font		
\large	large font		

表 3: 标准文档类中字号的绝对磅值

size	10pt (default)	11pt option	12pt option
\tiny	5pt	6pt	6pt
\scriptsize	7pt	8pt	8pt
\footnotesize	8pt	9pt	10pt
\small	9pt	10pt	11pt
\normalsize	10pt	11pt	12pt
\large	12pt	12pt	14pt
\Large	14pt	14pt	17pt
\LARGE	17pt	17pt	20pt
\huge	20pt	20pt	25pt
\Huge	25pt	25pt	25pt

表 4: 数学字体

\mathrm{...}	Roman Font
\mathbf{...}	**Boldface Font**
\mathsf{...}	Sans Serif Font
\mathtt{...}	Typewriter Font
\mathit{...}	*Italic Font*
\mathcal{...}	$\mathcal{CALLIGRAPHIC\ FONT}$
\mathnormal{...}	*Normal Font*

和字体命令相结合，花括号能够限定字体命令的作用范围。

示例代码：

```
He likes {\LARGE large and {\small small} letters}.
```

示例输出：

He likes large and small letters.

字体命令还有可能会改变行间距，但是只有当段落在字体命令作用范围内结束时这种情况才会发生。注意下面两个示例中 \par 的位置，以及它们行间距的不同。

示例代码 1：

```
{\Large Don't read this! It is not true.\\ You can believe me!\par}
```

示例输出 1：

Don’t read this! It is not true.

You can believe me!

示例代码 2：

```
{\Large This is not true either.\\ But remember I am a liar.}\par
```

示例输出 2：

This is not true either.
But remember I am a liar.

如果想改变一整段文字的字号的话，可以使用字体命令对应的字体环境，这样可以避免一些眼花缭乱的花括号。

示例代码：

```
\begin{Large}
    This is not true.\\ But then again, what is these days \ldots
\end{Large}
```

示例输出：

This is not true.
But then again, what is these days …

2.2 Danger, Will Robinson, Danger

正如本节开始时提到的那样，在文档中到处使用指定字体的命令是危险的，这不符合 LaTeX 的逻辑结构。如果想在文档中的许多地方用某种格式来排版特定的信息，我们应该使用 \newcommand 命令来定义一个“逻辑封装命令”来改变字体。

示例代码：

```
\newcommand{\oops}[1]{\textbf{#1}}
% in the document body:
Do not \oops{enter} this room, it's occupied by \oops{machines} of unknown purpose.
```

示例输出：

Do not **enter** this room, it's occupied by **machines** of unknown purpose.

需要注意告诉 LaTeX 去强调（*emphasize*）某些文字和使某些文字使用不同字体的区别。`\emph` 命令是可以知道上下文的，但是具体的字体设置命令是绝对的，不会因上下文的不同而发生变化。

示例代码：

```
\textit{You can also \emph{emphasize} text if it is set in italics,}
\textsf{in a \emph{sans-serif} font,}
\texttt{or in \emph{typewriter} style.}
```

示例输出：

You can also emphasize *text if it is set in italics,* in a *sans-serif* font, or in *typewriter* style.

2.3 建议

```
\underline{\textbf{Remember\Huge!}} \textit{The}
\textsf{M\textbf{\LARGE O} \texttt{R}\textsl{E}} fonts \Huge you
\tiny use \footnotesize \textbf{in} a \small \texttt{document},
\large \textit{the} \normalsize more \textsc{readable} and
\textsl{\textsf{beautiful} it bec\large o\Large m\LARGE e\huge s}.
```

Remember! *The* M**O** R*E* fonts you use **in** a document, *the* more READABLE and *beautiful it becomeS*.

3 间距（Spacing）

3.1 行间距（Line Spacing）

如果想要大一点的行间距，可以在文档的导言区加入 `\linespread{factor}` 命令。例如，使用 `\linespread{1.3}` 命令扩大到 1.5 倍行距，使用 `\linespread{1.6}` 扩大到双倍行距。通常行间距不会扩大，所以 *factor* 的值默认为 1。

需要注意，`\linespread` 命令的效果会很明显，所以轻易不要使用。如果非要用它改变行距的话，可以用下面的命令：

```
\setlength{\baselineskip}{1.5\baselineskip}
```

示例代码：

```
{\setlength{\baselineskip}{1.5\baselineskip}
This paragraph is typeset with the baseline skip set to 1.5 of what it
was before. Note the par command at the end of the paragraph.\par}

This paragraph has a clear purpose, it shows that after the curly brace
has been closed, everything is back to normal.
```

示例输出：

{\setlength{\baselineskip}{1.5\baselineskip}
This paragraph is typeset with the baseline skip set to 1.5 of what it was before. Note the par command at the end of the paragraph.\par}

This paragraph has a clear purpose, it shows that after the curly brace has been closed, everything is back to normal.

3.2 段落格式

在 LaTeX 中，有两个参数可以影响段落的布局。在导言区中加入以下命令可以改变这两个参数：

```
\setlength{\parindent}{0pt}
\setlength{\parskip}{1ex plus 0.5ex minus 0.2ex}
```

上面的两个命令将段落缩进设置为 0，并增加了段间距。

命令中的 `plus` 和 `minus` 会告诉 TeX，如果有必要的话，允许它根据指定的长度压缩或扩张段间距。

在欧洲大陆，经常是段首不缩进、段落之间留有空隙。但是需要留意的是，这也会对目录产生影响，目录的各行会变得更加稀松。为了避免这种情况，可以将上面的两个命令从导言区移动到文档正文的 `\tableofcontents` 命令之后，或者干脆就不要用这两个命令，因为大多数的专业书籍都是使用段首缩进、段落之间不留空隙的。

如果想让一个段首没有缩进的段落进行缩进，可以在段落开头添加 `\indent` 命令。显然，这个命令只有在 `\parindent` 没有设置为 0 时才会生效。

如果想创建一个段首不缩进的段落，可以在段落开头使用 `\noindent` 命令。这个命令通常用在文档起始处直接进入正文而没有使用 `\section` 命令的情况。

3.3 水平间距

LaTeX 会自动决定词句之间的间距。如果想要添加额外的水平间距，使用 `\hspace{length}` 命令。

如果在行首或行末也需要额外的间距，这时要换成 `\hspace*` 命令。*length* 是间距的长度，最简单的写法就是一个数值加一个单位。常用的单位见表 5。

示例代码：

```
This\hspace{1.5cm}is a space of 1.5 cm.
```

示例输出：

This is a space of 1.5 cm.

表 5: TEX 中常用的长度单位

mm	millimetre ≈ 1/25 inch
cm	centimetre = 10 mm
in	inch = 25.4 mm
pt	point ≈ 1/72 inch ≈ $\frac{1}{3}$ mm
em	approx width of an 'M' in the current font
ex	approx height of an 'x' in the current font

\stretch{n} 命令会产生一个特殊的有弹性的橡胶间距，它会伸展到将所在行的所有空隙占满。如果一行内使用了多个 \hspace{\stretch{n}} 命令，它们根据自己的伸展优先级来按比例分配这一行的所有空隙。

示例代码：

```
x\hspace{\stretch{1}}x\hspace{\stretch{3}}x
```

示例输出：

x x x

在文本中间添加水平间距时，我们希望间距的大小和当前的字号大小是一致的。这时就可以使用文本相关的单位 em 和 ex。

示例代码：

```
{\Large{}big\hspace{1em}y}\\
{\tiny{}tin\hspace{1em}y}
```

示例输出：

big y

tin y

3.4 垂直间距

段落之间、章节之间的间距是 LATEX 自动控制的。如果有必要，可以通过 \vspace{length} 命令来添加额外的垂直间距。

一般来说，这个命令应该放在两个空行之间。如果这个命令需要用在一页的开头或结尾，这时要换成 \vspace* 命令。

\stretch 命令配合 \pagebreak 命令，可以容易地排版出一页中的最后一行文字，或者用来使某一页的文字垂直居中。

示例代码：

```
Some text \ldots

\vspace{\stretch{1}}
This goes onto the last line of the page.\pagebreak
```

示例输出：

Some text …

This goes onto the last line of the page.

同一段落或表格内两行之间的额外间距可以通过 `\\[length]` 命令指定。

通过 `\bigskip` 和 `\smallskip` 命令可以插入一个预定义好的固定垂直间距。

4 页面布局（Page Layout）

LaTeX 2ε允许我们在 `\documentclass` 命令中指定纸张的大小。然后它会自动地选择合适的文本间距，但是有时候我们可能会对它的默认值不满意，这时我们可以通过 tools 宏集中的 `layouts` 宏包来自定义。图 1 展示了所有可以被自定义的参数。

等一下！ 当你欣喜万分，想“不如把这么窄的页面变宽”的时候，先冷静一下。LaTeX 中大多数东西之所以是那个样子，是有它们合理的理由的。

的确，和 MS Word 的页面相比，LaTeX 的页面显得有些窄。但是如果看一下我们喜欢的书籍，并数一数每一行大概有多少字符。然后你会发现一般每一行的字符数不会超过 66，同样地，LaTeX 默认的页面布局每一行一般也不超过 66 个字符。经验表明，当一行文字太长时，阅读起来会变得困难。这是因为一行文字如果太长，人们从这一行的末尾移动到下一行的开头会比较费劲。这也是为什么报纸要排版成多个竖栏。所以，当我们增加页面的宽度的时候，要考虑到这会使读者的阅读变得困难。

LaTeX 提供了两种命令来改变页面布局的参数。通常需要在文档的导言区使用这些命令：

```
\setlength{parameter}{length}       % 给某个参数设定一个具体的值
\addtolength{parameter}{length}     % 在某个参数值的基础上增加一定值
```

实际上，第二个命令比 `\setlength` 要常用，因为它是在已有设置的基础上改动的。

4.1 更有趣的长度

在写 LaTeX 文档的时候，最好尽可能地避免使用绝对长度。尽量使用一些别的页面元素的长度来替代绝对长度，例如我们想让图片的宽度铺满页面，就可以使用 `\textwidth` 命令。

下面的三个命令可以让我们决定文本字符串的宽度、高度、深度：

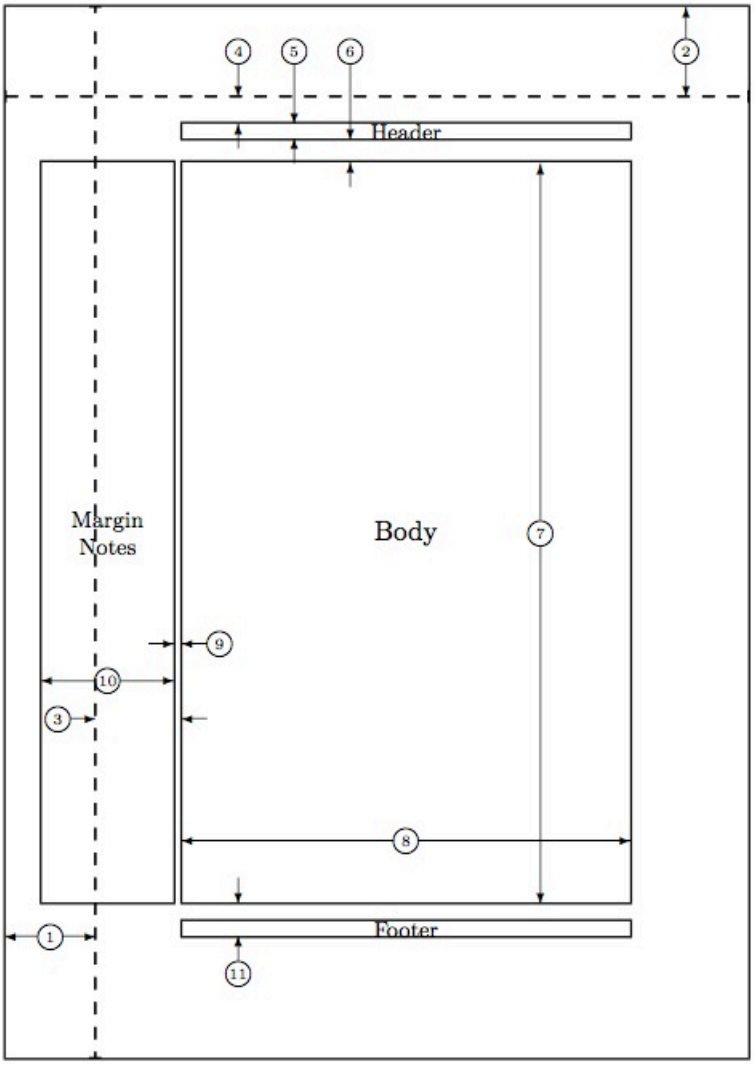
```
\settowidth{variable}{text}  % the width of the box
\settoheight{variable}{text} % the length between the baseline and the top of the box
\settodepth{variable}{text}  % the length between the baseline and the bottom of the box
```

示例代码：

```
\newenvironment{vardesc}[1]{
    \settowidth{\parindent}{#1:\ }
    \makebox[0pt][r]{#1:\ }}{}

\begin{displaymath}
a^2 + b^2 = c^2
\end{displaymath}

\begin{vardesc}{Where}$a$, $b$ -- are adjacent to the right angle of a right-angled triangle.
```




```
1   one inch + \hoffset          2   one inch + \voffset
3   \oddsidemargin = 22pt        4   \topmargin = 22pt
    or \evensidemargin
5   \headheight = 12pt           6   \headsep = 19pt
7   \textheight = 595pt          8   \textwidth = 360pt
9   \marginparsep = 7pt          10  \marginparwidth = 106pt
11  \footskip = 27pt                 \marginparpush = 5pt (not shown)
    \hoffset = 0pt                   \voffset = 0pt
    \paperwidth = 597pt              \paperheight = 845pt
```

图 1: *lshort* 一书中的页面布局参数。可以尝试 `layout` 宏包来打印我们自己文档的页面布局。

```
$c$ -- is the hypotenuse of the triangel and feels lonely

$d$ -- finally does not show up here at all. Isn't that puzzling?
\end{vardesc}
```

示例输出：

$$a^2 + b^2 = c^2$$

Where: a, b – are adjacent to the right angle of a right-angled triangle.

c – is the hypotenuse of the triangel and feels lonely

d – finally does not show up here at all. Isn't that puzzling?

5 盒子（Boxes）

LaTeX 通过盒子来构建页面。首先，每个字母都是一个小盒子，和其他的字母连接起来成为单词。单词又和单词连接在一起成为句子，但是单词和单词之间的连接是有弹性的，它们之间的空隙可以根据特定的情境来压缩或扩张。

这是一种非常简单的说法，但是却说明了盒子是 TeX 的操作单位。并不仅仅只有字母才能作为盒子，我们几乎可以将任何东西都放入盒子，包括一些其他的盒子。LaTeX 在处理时，就会把每个盒子作为一个字母。

之前已经遇到了一些盒子，例如 `tabular` 环境和 `\includegraphics` 命令，都会产生盒子。这意味着我们可以简单地将两个表格或两张图片并排排版，只要确保它们组合在一起后的宽度不超过文本宽度就行了。

我们还可以选定某一个段落，用 `\parbox[pos]{width}{text}` 命令或 `\begin{minipage}[pos]{width} text \e` 环境将其放入盒子。其中，*pos* 参数可以设置为 `c`、`t` 或 `b` 来控制盒子的垂直对齐方式。*width* 参数用于指定盒子的宽度。`minipage` 和 `\parbox` 的主要区别是，`parbox` 中不能使用所有的命令和环境，但是 `minipage` 中几乎可以任何东西。

`\mbox` 命令只处理水平对齐的内容。它只是把一些盒子打包成另一个盒子，也可以用来防止某两个词被 LaTeX 打断。由于盒子中可以嵌套盒子，这些水平方向的盒子有很高的灵活性。

```
\mbox{text}
\makebox[width][pos]{text}
```

width 决定了盒子外框的大小。除了长度表达式以外，还可以在 *width* 参数中使用 `\width`、`\height`、`\depth` 和 `\totalheight`，这些值都会从排版的文字获得。*pos* 参数可以是 `c`(center)、`l`(flushleft) 或 `s`(spread the text to fill the box)。

`\framebox` 和 `\makebox` 很像，但是它会在文本外面画出一个方框。

示例代码：

```
\makebox[\textwidth]{c e n t r a l} \par
\makebox[\textwidth][s]{s p r e a d} \par
\framebox[1.1\width]{Guess I'm framed now!} \par
\framebox[0.8\width][r]{Bummer, I am too wide} \par
```

```
\framebox[1cm][l]{never mind, so am I}
Can you read this?
```

示例输出：

c e n t r a l

s p r e a d

Guess I'm framed now!

Bummer, I am too wide

never mind, so am I

Can you read this?

前面是如何控制盒子的水平方向特性，而下面的命令可以设置盒子的垂直方向特性。

```
\raisebox{lift}[extend-above-baseline][extend-below-baseline]{text}
```

在前三个参数中，都可以使用 **\width**、**\height**、**\depth** 和 **\totalheight** 命令，从而对盒子中 *text* 产生影响。

示例代码：

```
\raisebox{0pt}[0pt][0pt]{\Large\textbf{Aaaa\raisebox{-0.3ex}{a}%
\raisebox{-0.7ex}{aa}%
\raisebox{-1.2ex}{r}%
\raisebox{-2.2ex}{g}%
\raisebox{-4.5ex}{h}}}
she shouted, but not even the next one in line noticed that something
terrible had happened to her.
```

Aaaaaaargh she shouted, but not even the next one in line noticed that something terrible had happened to her.

6 Rules

通常情况下我们用 **\rule**[lift]{width}{height} 来生成一个黑盒子，这在画水平线和垂直线时非常有用。注意在代码中，如果在行末紧跟着 %，表明这一行未结束，换行就不会产生空格。

示例代码：

```
\rule{3mm}{.1pt}%
\rule[-1mm]{5mm}{1cm}%
\rule{3mm}{.1pt}%
\rule[1mm]{1cm}{5mm}%
\rule{3mm}{.1pt}
```

示例输出：