第 4 章　特色功能

Du Ang
du2ang233@gmail.com

2017 年 7 月 23 日

目录

在编写大型文档的时候，LaTeX 还提供了一些像创建索引、管理参考文献等一些特色功能，详情见 *LaTeX Manual* 和 *The LaTeX Companion*。

1 包含 Encapsulated PostScript

借助 `figure` 和 `table` 环境，LaTeX 能够支持一些像图像、图形这种简单的浮动体。

在基本的 LaTeX 中或 LaTeX 的扩展包中，有很多种方法能够生成一些实际的图形。一种比较简单的方法是，通过一些专业软件生成这些图形，然后将它们包含到文档中。这里我们仅讨论使用 Encapsulated PostScript（EPS）来生成图形，因为这种方法非常简单，而且获得了广泛的应用。为了使用 EPS 格式的图片，必须要有 PostScript 打印机来输出。

D. P. Carlisle 开发的 `graphicx` 宏包提供了很多包含图片的命令，这个宏包属于“graphics”宏集。

假设现在的系统有可以输出 PostScript 打印机，也安装好了 `graphicx` 宏包，可以根据下面的步骤在文档中包含图片：

1. 通过画图程序输出 EPS 格式的图片。
2. 通过 `\usepackage[driver]{graphicx}` 命令，在导言区引入 `graphicx` 宏包。
   其中 *driver* 是 dvi 转 PostScript 的转换程序，最常用的一种叫 `dvips`。知道 *driver* 的名字后，`graphicx` 宏包就可以选择正确的方法来将图形信息插入到 `.dvi` 文件中，然后打印机就能理解它并且正确地包含 `.eps` 文件。
3. 在文档中使用 `\includegraphics[key=value, ...]{file}` 来包含 *file*。
   命令中的可选参数允许有多个，之间用逗号隔开。*key* 可以是 `width`、`height`、`angle`、`scale` 等参数，用于对包含的图形进行调整。

示例代码：

```
\begin{figure}
    \centering
    \includegraphics[angle=90, width=0.5\textwidth]{test}
    \caption{This is a test.}
\end{figure}
```

上面的代码包含了存储好的 `test.eps` 图片。图片旋转了 90 度，并且图片的宽度缩放到了标准图片的 0.5 倍。由于没有指定高度，所以默认宽高比是 1。宽度和高度也可以指定为具体的长度。

2 参考文献（Bibliography）

通过 `thebibliography` 环境来生成参考文献。每一个条目都以 `\bibitem[lable]{marker}` 开头，再通过 `\cite{marker}` 命令，就可以用来在文档中引用书籍、文章、论文等。

如果不指定 *label* 参数，所有的参考文献条目会自动编号。`\begin{thebibliography}` 命令后的参数用来定义应该为条目编号预留多少空隙。在下面的示例中，该参数为 `{99}`，表示所有的参考文献条目编号都不能比数字 99 更宽。

示例代码：

```
Partl~\cite{pa} has proposed that \ldots
\begin{thebibliography}{99}
\bibitem{pa} H.~Partl: \emph{German \TeX}, TUGboat Volume~9, Issue~1 (1988)
\end{thebibliography}
```

示例输出：Partl [1] has proposed that …

参考文献

[1] H. Partl: *German TEX*, TUGboat Volume 9, Issue 1 (1988)

对于更大型的项目，使用 BibTEX 是更好的选择。可以利用 BibTEX 建立一个参考文献数据库，然后再在文档中引用相关的文献。BibTEX 产生的参考文献格式是通过样式文件定义的，网上有很多现成的样式文件可供选择。

3 索引（Indexing）

索引是一个非常有用的功能，在很多书中都能看到它。LATEX 中有个 `makeindex` 程序，可以方便地建立索引。这里仅介绍基本的索引生成命令，更多内容见 *The LATEX Companion*。

为了建立索引，需要在导言区通过 `\usepackage{makeindex}` 命令来引入 `makeidx` 宏包，然后再在导言区中加入 `\makeindex` 命令来开启这项功能。

在正文中需要建立索引的地方，通过 `\index{key@formatted_entry}` 命令添加索引项。可选参数 *formatted_entry* 会出现在建立索引的地方；参数 *key* 用来排序。表 1 是索引项的写法示例。

表 1: 索引项写法示例

Example	**Index Entry**	**Comment**
`\index{hello}`	hello, 1	Plain entry
`\index{hello!Peter}`	Peter, 3	Subentry under ‘hello’
`\index{Sam@\textsl{Sam}}`	*Sam*, 2	Formatted entry
`\index{Lin@\textbf{Lin}}`	**Lin**, 7	Formatted entry
`\index{Kaese@K\"ase}`	**Käse**, 33	Formatted entry
`\index{ecole@\'ecole}`	école, 4	Formatted entry
`\index{Jenny|textbf}`	Jenny, **3**	Formatted page number
`\index{Joe|textit}`	Joe, *5*	Formatted page number

LATEX 在编译输入文件（`.tex` 文件）时，每一个 `\index` 命令都会把相应的索引项和当前页码写入和输入文件同名的 `.idx` 文件。`makeindex` 程序会对 `.idx` 文件进行处理，生成 `.ind` 文件。当再次编译输入文件时，遇到 `\printindex` 命令时会在文档中输出索引。

LATEX 2ε 中的 `showidx` 宏包将所有的索引项打印在相应的文本左侧，这在验证索引时十分有用。

4 自定义页眉页脚（Headers and Footers）

Piet van Oostrm 写的 fancyhdr 宏包提供了一些简单的命令，允许用户自定义文档的页眉页脚。

fancyhdr 宏包改善了页眉页脚的定义方式，允许我们将内容自由安置在页眉和页脚的左、中、右三个位置，还为页眉和页脚各加了一条横线。

fancyhdr 自定义了样式名称 fancy，使用 fancyhdr 宏包定义页眉页脚之前，通常先用 \pagestyle{fancy} 调用这个样式。在 fancyhdr 中定义页眉页脚的命令为：

```
\fancyhead[position]{...}
\fancyfoot[position]{...}
```

其中 *position* 参数为 L（左）/ C（中）/ R（右）以及与 O（奇数页）/ E（偶数页）字母的组合。

下面是使用 fancyhdr 的一个示例。它的效果为将章节标题放在和 headings 一致的位置，但使用加粗格式；页码都放在页脚正中；修改横线宽度，“去掉”页脚的横线。

示例代码：

```
% 导言区部分
\usepackage{fancyhdr}
\pagestyle{fancy}
\renewcommand{\chaptermark}[1]{\markboth{#1}{}}
\renewcommand{\sectionmark}[1]{\markright{\thesection\ #1}}
\fancyhf{} % 清空当前的页眉页脚
\fancyfoot[C]{\bfseries\thepage}
\fancyhead[LO]{\bfseries\rightmark}
\fancyhead[RE]{\bfseries\leftmark}
\renewcommand{\headrulewidth}{0.4pt} % 注意不用 \setlength
\renewcommand{\footrulewidth}{0pt}
```

5 Verbatim 宏包

前面我们已经接触过了 verbatim 环境，verbatim 宏包是在 verbatim 红包的基础上重新实现的，摆脱了原来的一些限制。

verbatim 宏包提供了 \verbatiminput{filename} 命令，允许我们将纯 ASCII 码文件以 verbatim 环境的样式插入文档。

verbatim 宏包属于“tools”宏集，所以大多数系统都已经预装了。

6 安装额外的宏包

大多数 LaTeX 发行版都已经预装了大量的样式宏包，但网上有更多，可以在 CTAN [1] 找到。

像 geometry、hyphenat 以及其他很多宏包，都由两个文件组成：.ins 文件和 .dtx 文件。有时会附加一个 readme.txt 文件，包含了对宏包的简要说明。

[1] http://www.ctan.org/

将宏包文件下载到计算机以后，需要执行以下步骤，一方面告诉 LaTeX 发行版我们安装了新宏包，另一方面获得宏包的文档：

1. 运行 `.ins` 文件，这会提取出 `.sty` 文件。
2. 将 `.sty` 文件移动到 LaTeX 发行版可以找到的地方。通常是 `.../localtexmf/tex/latex` 的子目录。
3. 刷新 LaTeX 发行版的文件名数据库。不同的 LaTeX 发行版对应的命令有所不同：TeXLive - `texhash`；web2c - `maktexlsr`；MiKTeX - `inittexmf --update-fndb`。

现在从 `.dtx` 文件中提取文档：

1. 运行 `.dtx` 文件，会生成一个 `.dvi` 文件。注意，可能需要编译多次才能保证交叉引用正确。
2. 检查是否产生了 `.idx` 文件。如果产生了该文件，继续执行下面的步骤；否则，说明文档没有索引，跳转到步骤 5。
3. 执行 `makeindex -s gind.ist name` 命令以产生索引。其中的 *name* 参数是不包含扩展名的主文件名。
4. 再次运行 `.dtx` 文件。
5. 通过 `.dvi` 文件生成 `.ps` 或者 `.pdf` 文件进行阅读。

有时候会看到一个 `.glo` （glossary）文件，这时需要在步骤 4 和步骤 5 之间执行 `makeindex -s gglo.ist name.gls name.glo`。还要确保在执行步骤 5 之前再运行一遍 `.dtx` 文件。

7 使用颜色

LaTeX 原生不支持颜色，它依赖 `color` 或者 `xcolor` 宏包。

7.1 颜色的表达方式

调用 `color` 或者 `xcolor` 宏包后，我们就可以使用如下命令切换颜色：

```
\color[color-mode]{code}
\color{color-name}
```

颜色的表达方式有两种，其一是使用色彩模型和色彩代码，代码用 0 sim 1 的数字代表成分的比例。`color` 宏包支持 `rgb`、`cmyk` 和 `gray` 模型，`xcolor` 支持更多的模型如 `hsb` 等。

示例代码：

```
\large\heiti
{\color[gray]{0.6} 60\% 灰色} \\
{\color[rgb]{0, 1, 1} 青色}
```

示例输出：

60% 灰色
青色

其二是直接用名称代表颜色，前提是已经定义好了颜色名称（没定义的话会报错）。
示例代码：

```
\large\heiti
{\color{red} 红色}
{\color{blue} 蓝色}
```

示例输出：

红色 蓝色

color 宏包仅定义了 8 种颜色名称，包括 black、red、green、blue、white、cyan、magenta、yellow。
xcolor 宏包补充了一些颜色，还包括了 darkgray、gray、lightgray、brown、olive、orange、lime、purple、teal、violet、pink，总共有 19 种颜色。
xcolor 还支持将颜色通过表达式混合或互补。
示例代码：

```
\large\heiti
{\color{red!40} 40\% 红色} \\
{\color{blue} 蓝色 \color{blue!50!black} 蓝黑 \color{black} 黑色} \\
{\color{-red} 红色的互补色}
```

示例输出：

40% 红色
蓝色 蓝黑 黑色
红色的互补色

我们还可以通过命令自定义颜色名称，注意这里的 *color-mode* 是必选参数：`\definecolor{color-name}{color-m`
如果调用 color 或 xcolor 宏包时指定 dvipsnames 选项，就有额外的 68 种颜色名称可用。xcolor 宏包还支持通过指定其它选项载入更多颜色名称。

7.2 带颜色的文本和盒子

原始的 `\color` 命令类似于字体命令 `\bfseries`，它使之后排版的内容全部指定的颜色，所以直接使用时通常要加花括号分组。color / xcolor 宏包都定义了一些方便用户使用的带颜色元素。
输入带颜色的文本可以用类似 `\textbf` 的命令：

```
\textcolor[color-mode]{code}{text}
\textcolor{color-name}{text}
```

以下命令构造一个带背景色的盒子，*material* 为盒子中的内容：

```
\colorbox[color-mode]{code}{material}
\colorbox{color-name}{material}
```

以下命令构造一个带有背景色和有色边框的盒子，*fcode* 或 *fcolor-name* 用于设置边框颜色：

```
\heiti
文字用 \textcolor{red}{红色} 强调 \\
\colorbox[gray]{0.95}{浅灰色背景} \\
\fcolorbox{blue}{yellow}{\textcolor{blue}{蓝色边框 + 文字，黄色背景}}
```

文字用 红色 强调
浅灰色背景
蓝色边框 + 文字，黄色背景

8 使用超链接

PDF 文档格式是现今最流行的电子文档格式，而电子文档格式最实用的需求之一就是链接功能。LaTeX 实现这一功能的是 hyperref 宏包。

8.1 hyperref 宏包

hyperref 宏包涉及到的链接遍布 LaTeX 的每一个角落——目录、引用、脚注、索引、参考文献等等都被封装成链接。但这也使得它与其它宏包的冲突机会大大增加，虽然宏包已经尽力解决各方面的兼容性，但仍不能面面俱到。为减少冲突的可能性，习惯上将 hyperref 宏包放在其它宏包之后调用。

hyperref 宏包提供了命令 \hypersetup 配置各种参数，或者也可以作为宏包选项，在调用宏包时指定：

```
\hypersetup{option1, option2, ...}
\usepackage[option1, option2, ...]{hyperref}
```

表 2: hyperref 宏包提供的参数设置

参数	默认值	含义
colorlinks=⟨*true*|*false*⟩	*false*	设置为 *true* 为链接文字带颜色，反之加上带颜色的边框
hidelinks		取消链接的颜色和边框
pdfborder={*n n n*}	0 0 1	超链接边框设置，设为 0 0 0 可取消边框
bookmark=⟨*true*|*false*⟩	*true*	是否生成书签
bookmarkopen=⟨*true*|*false*⟩	*false*	是否展开书签
bookmarknumbered=⟨*true*|*false*⟩	*false*	书签是否带章节编号
CJKbookmarks=⟨*true*|*false*⟩	*false*	使用 CJK 宏包 / GBK 编码排版中文时必须设定的参数，在第一次编译后需要将生成的 .out 文件用工具处理编码
unicode		使用 CJKutf8 宏包 / UTF-8 编码排版中文时必须设定的参数
pdftitle=⟨*string*⟩	空	标题
pdfauthor=⟨*string*⟩	空	作者
pdfsubject=⟨*string*⟩	空	主题
pdfkeywords=⟨*string*⟩	空	关键词
pdfstartview=⟨*Fit*|*FitH*|*FitV*⟩	*Fit*	设置 PDF 页面以适合页面/适合宽度/适合高度等方式显示，默认为适合页面

8.2 超链接

hyperref 宏包提供了直接书写超链接的命令，用于在 PDF 中生成 URL：

```
\url{url}
\nolinkurl{url}
```

\url 和 \nolinkurl 都生成可以点击的 URL，区别是前者有彩色，后者没有。在 \url 命令中作为参数的 URL 里，可直接输入如%、& 这样的特殊符号。

我们也可以像网页一样，把一段文字赋予其“超链接”的作用：\href{url}{text}

示例代码：

```
\url{http://wikipedia.org} \\
\nolinkurl{http://wikipedia.org} \\
\href{http://wikipedia.org}{Wiki}
```

示例输出：

http://wikipedia.org
http://wikipedia.org
Wiki

使用 hyperref 宏包后，文档中所有的引用、参考文献、索引等等都转换为超链接。用户也可以对某个 \label 命令定义的标签 *label* 作超链接（注意这里的 *label* 虽然是可选参数的形式，但通常是必填的）：\hyperref[label]{text}。

默认的超链接在文字外边加上一个带颜色的边框（在打印 PDF 时边框不会打印），可指定 *colorlinks* 参数修改为将文字本身加上颜色，或修改 *pdfborder* 参数调整边框宽度以“去掉”边框；*hidelinks* 参数则令超链接既不变色也不加边框。

示例代码：

```
\hypersetup{hidelinks}
\hypersetup{pdfborder={0 0 0}}
```

8.3 PDF 书签

hyperref 宏包另一个强大的功能是为 PDF 生成书签。对于章节命令 \chapter、\section 等，默认情况下会为 PDF 自动生成书签。和交叉引用、索引等类似，生成书签也需要多次编译源代码，第一次编译将书签记录写入 .out 文件，第二次编译才正确生成书签。

书签的一些属性见表 ??。在 latex + dvipdfmx 或 pdflatex 命令下使用 CTEX 或 CJK 宏集时，为了正确生成中文书签而不出现乱码，需要额外的设置，甚至繁琐的工序（这也是推荐使用 xelatex 命令处理中文的原因）。

hyperref 还提供了手动生成书签的命令：\pdfbookmark[level]{bookmark}{anchor}。其中，*bookmark* 为书签名称，*anchor* 为书签项使用的锚点（类似交叉引用的标签）。可选参数 *level* 为书签的层级，默认为 0。

章节命令里往往有 LATEX 命令甚至数学公式，而 PDF 书签是纯文本，对命令和公式的处理很困难，有出错的风险。hyperref 宏包已经为我们处理了许多常见命令，如 LATEX 和字体命令 \textbf 等，对于未被处理的命令或数学公式，就要在章节标题中使用如下命令，分别提供 LATEX 代码和 PDF 书签可用的纯文本：

```
\texorpdfstring{LaTeX code}{PDF bookmark text}
```

比如在章节名称里使用公式 $E = mc^2$，而书签使用字符 E=mc^2：

```
\section{质能公式 \texorpdfstring{$E=mc^2$}{E=mc\textasciicircum 2}}
```