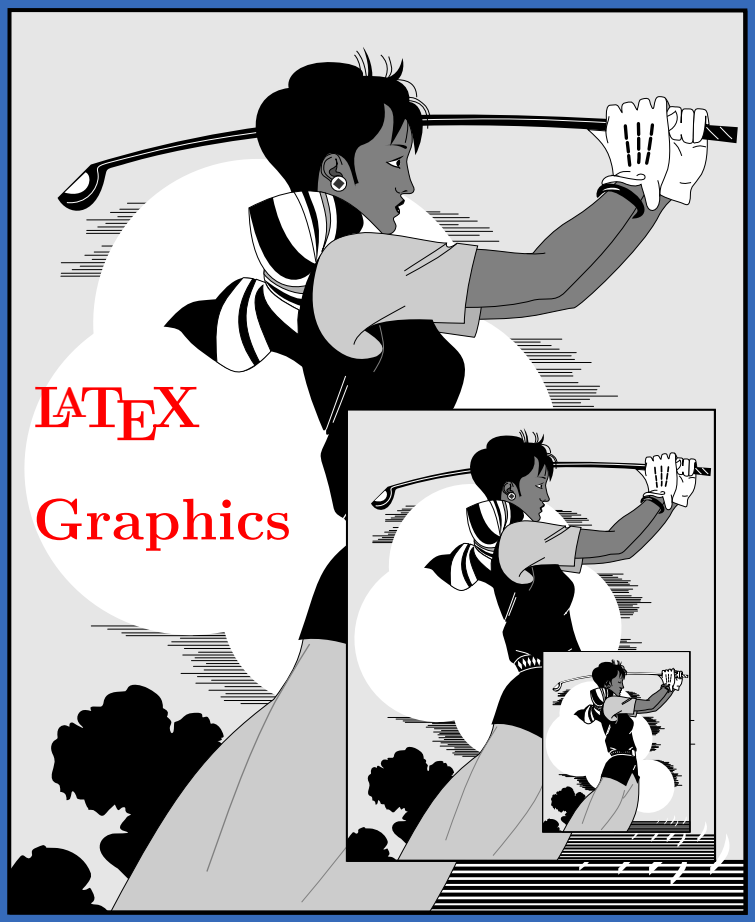
LaTeX 2ε



插图指南

Using Imported graphics in LaTeX and pdfLaTeX

Keith Reckdahl
epslatex at yahoo dot com

Version 3.0.1
January 12, 2006

译序

本文档是 Keith Reckdahl 所著 “Using Imported Graphics in LaTeX and pdfLaTeX” 的中文译本。

Keith Reckdahl 于 1997 年原著有文档 “Using Imported Graphics in LaTeX 2_ε” （版本 2.0，以下简称为 epslatex2），并且已经由王磊前辈在 2000 年将其翻译成《LaTeX 2_ε 插图指南》。该译本由于质量上乘，历史悠久，并且作为帮助文档附在 CTEX 套装内发行，因此在国内流传甚广、影响力甚大。

然而，也正是因为历史悠久，所以出现了很多与当前 LaTeX 插图技术脱节的地方。例如只讨论 latex+dvips 模式下 eps 图像格式的插入，以及使用 caption2 等过时宏包进行插图标题的处理等等。为此，原作者在 2006 年将 epslatex2 更新为版本 3.0.1（以下简称为 epslatex3）。虽然关于 LaTeX 插图这一话题有不少书籍和电子文档都有相关介绍，不过即便距今已十年有余，但英文版 epslatex3 仍可以说是最详尽的参考文档。而国内除了一些出版的书籍外，尚没有一份全面的专题文档介绍关于插图的各方面知识。这也正是翻译 epslatex3 的动机。

随着这些年 LaTeX 的发展，特别是相关宏包以及 XeTeX、LuaTeX等新编译引擎的出现和发展，2006 年版本的 epslatex3 也有不少已经过时的知识。因此在翻译过程中也加入了译者自己的一些修改。较大的改动有：

- 原文档第 7 节只讨论了 dvips 和 pdftex 两种引擎程序。这里加入了 dvipdfmx、xetex 和 luatex 等更多编译方式。
- 原文档 12.1 节提到了 overpic 宏包，本文档基于旧译本，在第 15.2 节加以详细介绍。
- 在第 16.3.1 节简要介绍了 eso-pic 宏包用法。
- 原文档的第 32 节使用 subfig 宏包处理子图标题，这里代之以介绍 subcaption 宏包。
- 第 34 节关于图文混排的内容来自于旧译本。

对于明显过时的章节将在行文处加以说明，但出于完整性考虑仍然翻译出来供参考。此外，行文各处的细节修改则通过脚注的方式给出。

本文档主要是在王磊前辈的译文基础上，结合英文版 epslatex3 对 epslatex2 的变动翻译而来。考虑到目前的 LaTeX 编译程序并非只有 pdfTeX，因此参照旧译本仍命名为《LaTeX 2_ε 插图指南》。继续使用或修改了旧译本中添加的内容已征得原译者许可。在翻译过程中还参考了刘海洋的著作《LaTeX 入门》以及其它相关宏包的文档，在此一并表示感谢。

盛文博
<wbsheng88@foxmail.com>
二零一七年八月

前言

在用 LaTeX 编写论文、书稿时，经常要遇到使用图形的情况。本书将讲述如何在 LaTeX 文件中插入图像以及其它一些相关问题。[1] 由于完全阅读本书需要较多的时间，如果想要查询某一特定的信息，那么可以通过目录和索引直接阅读相关内容。

插图的基本步骤是，首先确认宏包的导入

```
\usepackage{graphicx}
```

然后使用 `\includegraphics` 命令来插图

```
\includegraphics{file}
```

该命令的更多细节将在第 7 节介绍。

本文档分为如下的五个部分。

第一部分：背景介绍 这一部分介绍了一些有关的历史资料和基本的 LaTeX 术语。此外还包括

- Encapsulated PostScript (eps) 图形格式，eps 与 ps 文件的区别，以及将其它图形格式转为 eps 格式的方法。
- 通过 pdfTeX引擎可以直接导入的图片格式（jpeg, png, pdf, METAPOST）。
- 一些与图形有关的自由软件和共享软件。

第二部分：LaTeX 图形宏集 这一部分详细介绍了 LaTeX 2_ε 图形宏集中用于导入、缩放和旋转图形的命令。这部分涵盖了 LaTeX 2_ε 图形宏集文档的大部分内容 [1]。

第三部分：LaTeX 2_ε 图形命令的使用 这一部分介绍了如何使用 LaTeX 2_ε 图形宏包套件中的命令来引入、缩放和旋转图形。此外还讨论了以下三种情况：

- 在支持管道（例如 UNIX ）的系统中，使用 dvips 还可以实时地插入压缩的 EPS 图形或其它格式的图形（tiff, gif, jpeg, pict 等）。在其它的系统中，非 eps 格式图形必须先转换为 eps 才行。因为无论 LaTeX 还是 dvips 都没有内置解压缩和转换图像格式转换的功能，所以相应软件需要使用者提供。
- 由于许多图形应用程序只支持 ASCII 文本格式，psfrag 宏包可以将 eps 图形中的文字替换为 LaTeX 符号或数学表达式。
- 当一个 eps 图形被多次使用时（比如文字后面或页眉上的徽标），最后生成的 ps 文件会包含此 eps 图形的很多个副本。当所使用的图形不是位图格式时，可以通过定义 PostScript 命令来避免此图形被重复插入，从而减小得到的 eps 文件体积。

[1] 原文档涉及到图片的单词包括“figure”和“graphic”，前者指的是 LaTeX 文档中作为独立单元的图元素，而后者更偏向于来自于外部的图片文件及图片内容。本译文中将 figure 和 graphic 分别翻译为“图形”和“图像”。——译注

第四部分：`figure` 环境 将插入的图形放置于 `figure` 环境中有几个优点：在 `figure` 环境中的图形会被自动编号，从而可以引用或添加到目录中；置于该环境中的图形可以通过浮动实现较好的分页，所以可以很容易制作出具有专业水准的文稿。除了介绍 `figure` 环境的一般信息外，这一部分还讲述了如下一些和图形有关的话题：

- 怎样自定义 `figure` 环境，例如怎样调整图形的位置、图形周围的距离、标题的距离，以及在图形与文本之间加入分隔线等。还可以自定义标题的格式，改变标题的样式、宽度和字体等。
- 怎样创建边注图形和伸入边注位置的宽图形。
- 怎样在纵向页面版式的文档中加入横向图形。
- 怎样将标题放置于图形的两边而不是上下。
- 对于双面排版的文档，怎样确保一幅图形放置于奇数页或偶数页。还有，怎样确保两幅图形都出现在同一对开页面上。
- 怎样得到带框的图形。

第五部分：复杂图形 这一部分介绍如何创建较为复杂的图形，用以同时包含多个图像。

- 怎样形成并列的图形、图像和子图。
- 怎样在一个浮动体内将一个表格放置于图形之后。
- 怎样堆叠多行图形。
- 怎样创建跨页的连续图形。

如何得到本文档

本书的 `pdf` 格式和 `ps` 格式的英文原版网络位置为[2]

CTAN/info/epslatex/english/epslatex.pdf
CTAN/info/epslatex/english/epslatex.ps

本文档版本 2.0 的法文版本由 Jean-Pierre Drucbert 完成，其 `pdf` 和 `ps` 文件可以从如下位置获得

CTAN/info/epslatex/french/fepslatex.pdf
CTAN/info/epslatex/french/fepslatex.ps

完整的 CTAN (Comprehensive TEX Archive Network) 映像站点可以从 `https://ctan.org/mirrors` 获得。

致谢

略。

[2] 此段有删改。——译注

许可协议

目录

第一部分　背景知识

1　简介

历史渊源

当 Knuth 编写 TEX 的时候，还没有 PostScript/eps、jpeg、gif 以及其它图像格式。因此 dvi 文件对于图像导入并没有直接的支持。不过，TEX 允许 dvi 文件中包含 \special 命令，通过它可以向调用 dvi 文件的程序传递命令。因此，凡是调用 dvi 的程序支持的图像格式，TEX 和 LATEX 都能够导入。

过去的许多年来，dvi 通常都被转为 PostScript 格式，进而标准的插图格式是 Encapsulated PostScript（eps）图像，因为它是 PostScript 语言的一个子集。在 LATEX 中插入 eps 图像最初通过低层命令 \special 来完成。为了使插图更加方便并且更具有可移植性，专门为 LATEX2.09 设计了两个高层的宏包 epsf 和 psfig。epsf 提供了 \epsfbox 命令来插入图片，此外另有三个命令来控制图片的缩放。在 psfig 中，\psfig 命令除了用来插入图片，还有缩放和旋转功能。不过，尽管 psfig 的句法更受欢迎，它的代码却没有 epsf 健壮。于是作为这两个宏包的结合的产物，epsfig 宏包使用 psfig 的句法和大部分 epsf 的健壮代码。不过，epsfig 仍然使用了一些不健壮的 psfig 代码。

LATEX 图形宏集

随着 1994 年 LATEX 2ε 的发布，LATEX3 小组认识到在 LATEX 插图中存在的一些普遍问题。通过努力，他们开发出了“LATEX 图形宏集”，[3] 并全部重写了其中的命令。比起其它的插图命令，他们的命令更高效、更健壮、更具有可移植性。

LATEX 图形宏集包括“标准”的 graphics 宏包和“扩展”的 graphicx 宏包。这两个宏包都有一个 \includegraphis 命令，不过版本不同。类似 psfig 的句法，graphicx 版的 \includegraphis 采用“命名参数（named arguments）”。这使用起来比较简单方便，却违反了 LATEX 关于可选参数放置位置的语法指南方针。作为折中方案，就有了两种版本的 \includegraphics。其中，graphics 的版本遵从 LATEX 的语法规则，而 graphicx 的版本则采用更为简便的命名参数。graphicx 版本的 \includegraphics 支持图形的缩放和旋转，而 graphics 版本的 \includegraphics 则必须被置于 \scalebox 或 \rotatebox 才能达到同样的效果。

本文档使用 graphicx 宏包，因为它的句法比 graphics 宏包更加简便易用。其实这两个宏包具有相同的功能，本文档中的例子同样可以用 graphics 宏包完成，只不过相应的命令有些笨拙和缺少一点效率。关于这两个宏包详细的说明可参见 LATEX 图形宏集的文档 [1]。

出于向后兼容性，LATEX 图形宏包套件中也提供了一个 epsfig 宏包，用以替代之前 LATEX 2ε 的 epsfig。新的 epsfig 定义了 \epsfbox、\psfig 和 \epsfig 等命令。不过它们只是调用 \includegraphics 命令的简单封装。由于这些封装效率没有直接使用 \includegraphics 命令高，因此，该 epsfig 宏包应当只用于旧文档。在编写新文档时要用 \includegraphics。

非 eps 图形

除了改进 eps 图片的部分，LATEX 图形宏集还处理非 eps 图像格式的插图问题，例如 jpeg 和 gif 等。由于 dvi 到 ps 的转换程序一般不直接支持大部分的非 eps 格式，向 PostScript 文档中插入这些图像之前相关图像必须首先转成 eps 格式。尽管这种图像格式的预处理通常是最佳办法，不过，LATEX 图形宏集还提供了另一种选择：dvi 转 ps 时自动地实时转换图像格式。第 6.2 节（18 页）介绍了一下图像转换程序，第 14 节（46 页）

[3] 已经有 LATEX 图形宏集的 plain TEX 版本，相关文件请见 CTAN/macros/generic/graphics/

讲述了如何在 dvi 转 ps 时使用非 eps 格式的图像。

pdfTeX

过去，PostScript 通常是 LaTeX 文档的最终格式，整个编译过程有两个步骤：(1) 使用 LaTeX 生成 dvi 文件，(2) 使用一个 dvi 到 ps 的转换程序（例如 dvips）来生成 PostScript 文件。之后，随着 Adobe 公司的 pdf 格式开始流行，编译过程又加了第三步：(3) 使用 Ghostscript[4]、Adobe Acrobat[5] 或者 PStill[6] 等工具将 PostScript 文件转成 pdf 文件。

然而，这种三步骤过程 LaTeX-dvips-ghostscript 不仅很繁琐，而且很难实现一些 pdf 特性，比如超链接。为了改进这一点，Hàn Thế Thành 写了一个工具叫做 TeX2PDF，这一工具修改了 TeX 引擎，可以直接从 TeX 生成 pdf。TeX2PDF 最终重命名为 pdfTeX，并且在许多志愿者的帮助下（托高德纳的福）进行扩展，进而实现了 TeX 的全部排版功能。pdfTeX从名称上看来输出的是 pdf，不过它也能输出 dvi 格式，并且与原先 TeX 引擎的输出结果相同。

如同 latex 命令使用 TeX 引擎处理 LaTeX 文档并生成 dvi 文件，[7] pdflatex 命令使用 pdfTeX 引擎处理 LaTeX 文档，并直接生成 pdf 文件。

pdfTeX 和图像

pdfTeX 的一个重要特性就是原生支持许多图像格式：jpeg、png、pdf、METAPOST。尽管老版本的 pdfTeX 还支持 tiff 文件，目前的 pdfTeX 版本不支持 tiff。

另外要注意的是，pdfTeX 不能直接导入 eps 文件[8]，用户需要用 epstopdf 等程序将 eps 文件转成 pdf 格式，不过这样就不能直接使用 PSfrag 宏包（见第 15.1 节，49 页）。

其它编译引擎

目前[9]，除了 latex+dvips 和 pdflatex 之外，还有几种常见的编译方式：

dvipdfmx 可以从 dvi 文件生成 pdf 文件。

xelatex 可以直接生成 pdf 文件。支持 UTF-8 编码和使用各种字体。

lualatex 作为 pdflatex 的后继引擎，引入了 Lua 动态编程语言。可以直接生成 pdf 文件。支持 UTF-8 编码和使用各种字体。

出于技术上的先进性、稳定性以及效率和兼容性考虑，一般而言推荐使用 xelatex 编译方式。特别是对于中文文档，这样可以直接使用系统字体，避免了繁琐的中文配置过程。而对于英文文档，也可以考虑使用 pdflatex。

以上几种编译方式都原生支持 eps、pdf、jpg、png 等各种图片格式。不过这样就不能直接使用 PSfrag 宏包（见第 15.1 节，49 页），因为该宏包要求输出格式是 PostScript。

2 LaTeX 术语

任何 LaTeX 对象（字符，图形等）都把**盒子**作为单位（[7, 103 页]）。每个盒子在它的左侧均有一个**参考点**（*Reference point*）。盒子的**基线**（*baseline*，见图 1）是一条通过参考点的水平线。当 LaTeX 从左到右排列文本时，每一字符的参考点排成一条水平直线，称为**当前基线**（*current baseline*），并使它与字符的基线对齐。LaTeX 也用同样的方法来处理图形和其它对象，每个对象的参考点都被放置于当前基线上。

[4] 自由软件，见第 6.1 节（18 页）。

[5] 商业软件，见 www.adobe.com

[6] 共享软件，见 www.pstill.com

[7] 现代的 TeX 发行版本中，latex 命令使用的也是 pdfTeX 引擎。——译注

[8] pdfTeX 可以导入由 PurifyEPS 处理的 eps 文件，见第 5.5 节（17 页）。

[9] 此段由译者添加

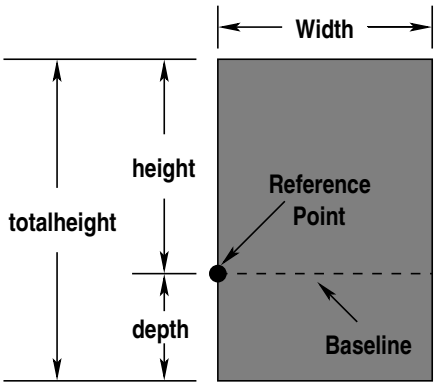


图 1: LaTeX 盒子示例

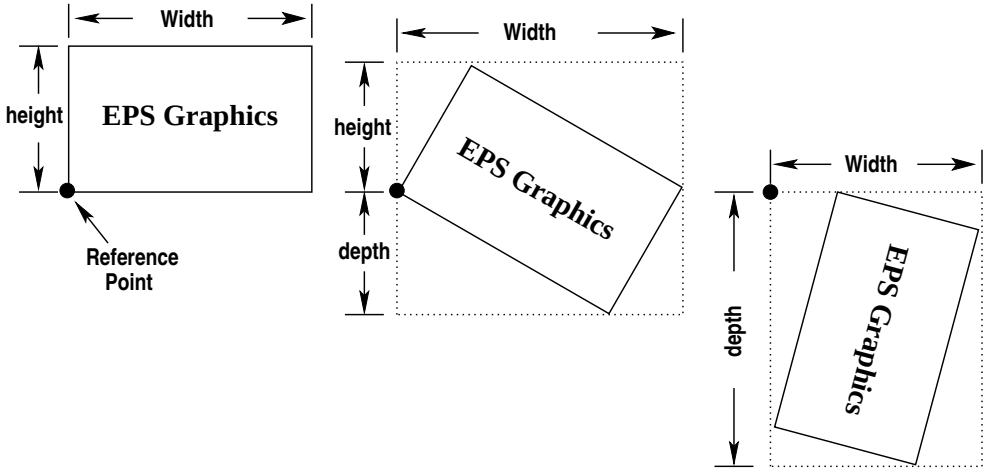


图 2: LaTeX 盒子的旋转示例

每一个盒子的大小由三个长度决定：高度（*height*）、深度（*depth*）和宽度（*width*）。高度是参考点到盒子顶部的距离，深度是参考点到盒子底部的距离，宽度则是盒子的宽度。而整体高度（*totalheight*）定义为盒子底部到顶部的距离，即：整体高度 = 高度 + 深度。

未旋转图形的参考点是它的左下角（见图 2 的左边的盒子），它的深度为零，高度就等于全部高度。图 2 中间的盒子则是将图形旋转后，它的高度就不等于全部高度了。右边的盒子则展示可将图形旋转使其高度为零。

3 Encapsulated PostScript

PostScript 语言能够用来描述图形和文本。它既可在传统的 PostScript （`ps`）文件中来描述多页文档，也用于 Encapsulated PostScript （`eps`）文件中来描述插入文档的图像。`ps` 文件和 `eps` 文件的主要区别在于：

- `eps` 文件只能使用部分特定的 PostScript 操作符。
- `eps` 文件必须含有一个 BoundingBox 行来确定 `eps` 图像的大小。

3.1 禁止使用的 PostScript 操作符

由于 eps 图形需要和其它对象一起共享页面，所以 EPS 文件中不能使用页面操作，例如选择页面大小（a4 或 letter）和清除整个页面（erasepage）等命令。下面是不能在 eps 文件中使用的 PostScript 操作符：

a3	a4	a5	banddevice	clear
cleardictstack	copypage	erasepage	exitserver	framedevice
grestoreall	initclip	initgraphics	initmatrix	letter
legal	note	prenderbands	quit	renderbands
setdevice	setglobal	setpagedevice	setpageparams	setsccbatch
setshared	startjob	stop		

尽管下列 PostScript 操作符可以在 eps 文件中使用，但是不适当地使用它们极易导致错误。

nulldevice	setcolortransfer	setgstate	sethalftone
setmatrix	setscreen	settransfer	undefinedfont

上面的一些操作符可能会使 dvi 到 ps 的转换失败，另一些则可能导致像图像位置错误、消失或者闪烁等奇怪的问题。因为这些操作符绝大部分不会影响到 PostScript 的堆栈，所以在大多数情况下，简单地将这些招致问题的操作符删除就可解决问题。其它的情形则需要更为复杂的 PostScipt 编程知识。

3.2 EPS BoundingBox

习惯上，PostScript 文件的第一行标明了该文件的 PostScript 类型，接下来的几行是被称为 *header* 或 *preamble* 的注释行（PostScript 的注释符也是 %）。其中一行定义了 BoundingBox，包括四个整数值，分别代表:

1. BoundingBox 的左下角的 x 坐标。
2. BoundingBox 的左下角的 y 坐标。
3. BoundingBox 的右上角的 x 坐标。
4. BoundingBox 的右上角的 y 坐标。

例如，一个由 gnuplot 程序生成的 eps 文件的前五行为

```
%!PS-Adobe-2.0 EPSF-2.0
%%Creator: gnuplot
%%DocumentFonts: Times-Roman
%%BoundingBox: 50 50 410 302
%%EndComments
```

这个 gnuplot 生成的 eps 图像的左下角的坐标是 $(50, 50)$，右上角的坐标是 $(410, 302)$。这里坐标的单位是 PostScript 点，一点等于 $1/72$ 英寸。这样上面的这幅图的自然宽度为 5 英寸，相应的自然高度为 3.5 英寸。需要注意的是 PostScript 点要比 TEX 点（等于 $1/72.27$ 英寸）稍大。在 TEX 和 LATEX 中，PostScript 点被称为“大点”（“big points”）或简称 bp，而 TEX 点被称为 “points” 或简称 pt。

3.3 将 PS 转换为 EPS

单页的 PostScript 文件，如果没有包含不适当的命令的话，可用下述方法之一加上 BoundingBox 行并转为 `eps` 文件。**由于这些方法都不检查非法的 PostScript 操作符，所以只有在被转换的 PostScript 文件本身不含有那些被禁制使用的操作符的情况下，才能得到正确的 `eps` 文件。**

1. 最方便的是用 GhostScript 里带的 ps2epsi 工具（见第 6.1 节，18 页）。它可以读入 PostScript 文件，计算 BoundingBox 的参数，然后生成一个含有 PostScript 图像的 `eps` 文件。

   最终得到的 `eps` 文件是 `epsi` 格式，即它在文件的开始部分带有一个低分辨率的 Interchange 预览位图。因为这个预览位图是 ASCII 编码的，所以不会造成像第 4.1 节的 `bufsize` 错误。不过，`epsi` 预览会增加文件体积。

2. 另外一种通过 Ghostscript 计算 BoundingBox 参数的方法是使用 epstool 工具。该程序适用于 Unix、DOS、Windows 和 OS/2 系统，获取链接为

   `http://www.cs.wisc.edu/~ghost/gsview/epstool.htm`

   例如，如下命令

   ```
   epstool --copy --bbox file1.eps file2.eps
   ```

   分析 `file1.eps` 的内容来决定正确的 BoundingBox，然后将带有计算后的 BoundingBox 值的 `file1.eps` 内容复制给 `file2.eps`。epstool 工具还可以用于在 `eps` 文件内添加 `tiff`、`wmf` 以及 `epsi` 预览位图，或者用于从 `eps` 文件中提取预览位图。

3. 此外，还有一种办法是，计算 BoundingBox 的参数，然后把它加到 PostScript 文件中或作为插图命令的参数（比如用 `\includegraphics` 的 `bb` 选项）。计算 BoundingBox 参数的方法有以下几种:

   (a) 用 Ghostview 或 GSview 将 PostScript 图形打开。当鼠标在图形上移动时就会显示相应的坐标（以页面的左下角为参照点）。记下图形的左下角和右上角的坐标就可确定它的 BoundingBox。

   (b) 将 PostScript 图形打印一份，测量页面的左下角到图像的左下角的水平和竖直距离（以英寸为单位），然后乘以 72 就可以得到 BoundingBox 的左下角坐标。类似地，测量页面左下角到图像右上角的水平和竖直距离就可以得到 BoundingBox 的右上角坐标。

   (c) `bbfig` 脚本使用 PostScript 打印机来计算 BoundingBox。`bbfig` 会在 PostScript 文件开头添加一些 PostScript 命令并送往 PostScript 打印机。在打印机处，添加的 PostScript 命令会计算 BoundingBox 坐标，然后将结果打印在 PostScript 图像上面。`bbfig` 脚本程序的地址为

   CTAN/support/bbfig/

3.4 修正非标准的 EPS 文件

一些应用程序（例如 Mathematica 和 FrameMaker ）会生成非标准的 eps 文件，不能在 LaTeX 等其它程序上使用。有一些应用程序会根据它们自己的偏好给 PostScript 加入了额外特性，还有一些应用程序生成非常糟糕的 PostScript 代码。通常来说，这些非标准的 PostScript 可以由软件公司自己或者精通 PostScript 的用户提供的脚本来修正。检索软件供应商的网页或者搜索相关软件的网络新闻组来获取相关信息。

4 LaTeX 怎样使用 EPS 图

当处理 dvips 类型文件时，LaTeX 和 dvi 到 ps 的转换程序都会使用 eps 文件。

1. LaTeX 通过读取 eps 文件中的 BoundingBox 行来决定为 eps 图形保留多大的空间。
2. dvi 到 ps 的转换程序读取 eps 文件并把它插入到生成的 ps 文件中。

需要说明的几种情形：

- 如果在插图命令中给定了 BoundingBox 的值（例如，使用了 \includegraphics 的 bb 选项），LaTeX 将不会再读取 eps 文件。事实上，当运行 LaTeX 时，eps 文件甚至都不必存在。
- 由于 TeX 不能读取非 ASCII 文件，也不能生成其它的程序，所以 LaTeX 不能从压缩的 eps 文件或其它非 eps 图像文件中得到 BoundingBox 的信息。在这种情况下，BoundingBox 参数值必须由插图命令给定（例如 \includegraphics 命令的 bb 选项）或者存放在一个未压缩的文本文件中（见第 14 节，46 页）。
- eps 图像并没有包含在 dvi 文件中。当 dvi 转换为 ps 时，eps 文件必须存在。因此，所有用到的 eps 文件必须和 dvi 文件放在一起。
- 一些 dvi 浏览器可能不支持显示 eps 图像。这时，为了方便使用者对图像进行定位，一般会将图形的 BoundingBox 用一个方框显示出来。

4.1 行缓冲区溢出

TeX 在读取 ASCII 文件时是每次从中读取一行，然后把它放到自己的行缓冲区里。TeX 的行缓冲区大约有 3000 个字符长度。如果 eps 文件中有一行的长度超过了行缓冲区的长度，就会产生如下的错误讯息：

```
Unable to read an entire line--bufsize=3000.
Please ask a wizard to enlarge me.
```

因为 eps 很少有一行长度超过 3000 个字符的情形，所以产生行缓冲区溢出错误的原因可能有两种：

1. **EPS 文件中有一个长的二进制预览图**

   有些应用程序生成的 eps 文件在开始部分放置了一个二进制的预览图，这样就可使得像 dvi 浏览器等一些不能解释 PostScript 的软件也可来显示 eps 图形。目前有少数与 TeX 有关的软件使用这种方法。

如果这个二进制预览图比行缓冲区小，\includegraphics 将会略过它[10]。但是，如果这个二进制的预览图比行缓冲区大的话，就会发生行缓冲区溢出的错误。对于该问题有两种解决办法：

(a) 如果不需要预览图，可以用文本编辑器将它删掉或在一开始用应用程序生成 eps 图像时就选择不要预览图。

(b) 因为 LaTeX 读取 eps 文件的唯一目的就是取得 BoundingBox 参数值大小,、因此，如果插图命令中给出 BoundingBox 的值（如在 \includegraphics 中使用 bb 选项），那么 LaTeX 就不会读取 eps 文件。

2. **eps 文件中的分行符在不适当的传输中被损坏**

这里所谈到的问题不会在大部分最近的 TeX 发行版本中出现，因为这些软件中的 TeX 程序都会正确的识别所有的分行符。

不同的操作系统平台使用不同分行符。Unix 使用换行符 LF（^J），Macintosh 使用回车符 CR（^M），而 DOS/Windows 则使用回车符加换行符（^M^J）。比如一个 eps 文件从 Macintosh 机上用二进制方式传输到 Unix 机上，那么 Unix 机上的 TeX 会因找不到分行符 ^J 而把整个文件作为一行，导致行缓冲区溢出的错误。

如果 eps 文件中不含有二进制的部分（如预览图和嵌入的图形），以文本方式传输就可以解决这一问题。然而，带有二进制部分的 eps 文件必须用二进制方式传输，否则文本方式传输会损坏二进制部分。由于二进制传输方式不会翻译分行符，这时就需要在插图命令中给出 BoundingBox 的值来解决（例如 \includegraphics 命令的 bb 选项）。

5 PDF 图像

正如之前提到的，pdfTeX 可以直接导入 pdf、png、jpeg 和 METAPOST 图像格式。这一节将简要介绍这些格式。相关的pdfTeX 插图命令将在第 7 节（22 页）中介绍。

5.1 JPEG

jpeg 是由联合图像专家组（Joint Photographic Experts Group , JPEG）委员会

http://www.jpeg.org/

授权的压缩标准。jpeg 格式是一种位图的压缩标准，它采用了有损压缩格式[11]。特别地，压缩过程不会保持线条和锋锐的边缘。所以不太适用于线条描绘和带有锋锐元素的图片。

5.2 PNG

过去很多年 gif 一直是图标和其它线条画的位图压缩标准，这是由于它采用的无损 LZW 压缩不会使得锋锐的边缘失真。由于 Unisys 公司对于其 LZW 专利的强制执行，加上 gif 的一些技术局限（例如 256 色的限制），这些因素促使了一个小组开发可移植网络图形（Portable Network Graphics , PNG ）格式。该小组后来称为 png 开发组

[10] 注意，\psfig 和其它一些过时的图像命令则不会忽略二进制预览。

[11] 有损压缩意味着压缩过程中损失了数据。也就是说，解压缩一个有损压缩过的位图不会得到原始图片。反之，无损压缩过程中不会有数据损失，所以解压一个无损压缩过的位图会生成原始图片。

http://www.libpng.org/pub/png/

与 GIF 一样，PNG 也采用无损压缩，因此适合于线条画。尽管 png 可以用于任何位图文件，不过对于摄影照片和其它一些没有锋锐边缘的位图来说，jpeg 有损压缩通常效果更好（“更好”指的是生成更小的文件体积，同时对于人眼观察没有失真）。

5.3 PDF

Adobe 的便携式文档格式（Portable Document Format, pdf ）与它的 Adobe 成员 PostScript 有很多相似之处。与 PostScript 一样，pdf 可以包含文本、矢量图和位图。一个 pdf 文件可以包含整个文档，也可以仅仅包含一个图形（类似于 eps ）。

pdf 不仅仅是 pdfTEX 的主要输出格式，也是 pdfTEX 插图的最普遍方法。许多图像程序允许它们的图像直接保存为 pdf 格式。没有直接 pdf 输出的程序可以转而输出 eps 矢量图，后者通过 epstopdf 转换程序可以很容易地转成 pdf 矢量图。epstopdf 可以从 CTAN 获得，在 Windows 系统下是一个可执行程序，在 Unix/Linux 以及 MaxOSX 等其它系统上则是一个 perl 脚本

CTAN/support/epstopdf/

5.4 METAPOST

METAPOST 是由 John Hobby 编写的绘图语言。它基于高德纳的 METAFONT，但增加了 PostScript 输出的功能。关于 METAPOST 的信息可以从以下网址获得：

http://www.tug.org/metapost.html

相关文档可参考 [6]。

METAPOST可以用于 dvips 类型的 LATEX 文档，也可以直接用于 pdfLATEX 文档[12]。

下面的步骤使用 METAPOST（mpost ）所带的 pstoedit 工具将名为 graphic.eps 的 eps 文件转为名为 graphic.mps 的 METAPOST 文件：

```
pstoedit -f mpost graphic.eps graphic.mp
mpost graphic.mp
rename graphic.1 graphic.mps
```

5.5 PurifyEPS

Scott Pakin 的 purifyeps 工具能够将很多（但不是全部）eps 转成“净化版本”，从而可以直接被 LATEX 和 pdfLATEX 读取。为了运行 purifyeps 你需要下列工具：

PurifyEPS 可以从CTAN/support/purifyeps/ 获得。

Perl 可以从 http://www.cpan.org 获得。

pstoedit 可以从 http://www.pstoedit.net/pstoedit 获得。

mpost 来自于任何包含 METAPOST 的 LATEX 发行版本。

[12] pdfLATEX 实际上使用 Hans Hagen 开发的 ConTEXt 代码将 METAPOST 图像实时转成 pdf，不过在用户层面上是很简明的。

6 图像软件

[本节介绍的部分软件可能已经过时，仅供参考。——译注]

6.1 Ghostscript

Ghostscript 是一个 PostScript/pdf 解释器，它可以运行在大多数操作系统平台上，并且是自由发布的[13]。通过 Ghostscript，PostScript、eps 和 pdf 文件可以在屏幕显示，也可以用 Postscript 和非 PostScript 打印机来打印。AFPL Ghostscript 可以从 Ghostscript 官网获得：

http://ghostscript.com/

其中包含了预编译的 Windows、DOS、OS/2 和 Macintosh 可执行文件，以及在 Unix/VMS 下可用的源代码。此外还有一些 Ghostscript 的图形界面软件，例如 GSview、Ghostview、GV 等，可以更方便地浏览 PostScript。

Ghostscript 中还带有一些有用的工具，如 ps2pdf 等，可利用 Ghostscript 来转换图形，打印、预览 PostScript 文件。详细的使用说明可参考 Ghostscript 所带的使用说明文件。

6.2 图像格式转换工具

下面列出的一些免费软件和共享软件可以用来转换图像格式。在 dvips 模式文件中，这些程序可以将非 eps 图像转成 eps。在 pdfLaTeX 文档中，这些程序可以将图像转成可以支持的格式（pdf、png、jpeg）。其中一些程序还提供命令行方式，这样就可以实时转换图像，具体见第 47 页的 14.2 节

ImageMagick ImageMagick 是一个开放源代码的自由软件，适用于创建、编辑和合并位图图片。它可以读取、转换和写入各种格式的图片。可以进行的操作包括：剪裁图片、修改颜色或是应用各种效果，旋转和合并图片，添加、拉伸和旋转文字、线条、多边形、椭圆和贝塞尔曲线。

例如，当 ImageMagick 的 convert 命令添加到系统的环境变量后，如下命令会将 file.jpg 存为 eps 版本。

```
convert file.jpg file.eps
```

使用通配符可以批量转换文件，例如

```
convert *.gif image.png
```

会将当前路径下的所有 gif 文件转为 png 版本，并保存为

```
images-0.png
images-1.png
...
```

[13] 尽管 AFPL Ghostscript （之前叫做 Aladdin Ghostscript ）是免费发布的，但却不是公有领域。它受版权保护，有一些限制，例如不允许商业发布等。当 Aladdin Ghostscript 的版本停止了大约一年之后，Aladdin 将其以“GNU Ghostscript”发布，使用的是限制性不那么强的 GNU Public License。

想要将 png 文件名保存成原始的 gif 文件名则比较复杂，需要写一个 shell 脚本或者 Windows 批处理文件。

ImageMagick 可以运行在所有主流的操作系统上。可执行程序和相关信息可以从下面的网址下载

http://www.imagemagick.org/

GraphicsMagick ImageMagick 的接口不时发生变化，这会导致调用 ImageMagick 的代码发生不兼容性。作为 ImageMagick 5.2.2 的一个分支，GraphisMagick 项目于 2002 年 11 月开始开发，旨在提供一套具有稳定接口的图像转换工具，它的侧重点在于修复 bugs 而不是添加新特性。

GraphicsMagick 可以在 Unix/Linux、Cygwin、MacOSX 和 Windows 系统上运行。可执行程序和源代码可以从如下网址下载

http://www.graphicsmagick.org/

NetPBM NetPBM 是现在已经不维护的 PBMPLUS 程序包的自由开源版本。NetPBM 是一个关于图像操作的工具箱，包括在各种类型的格式之间相互转换。该程序包内有超过 220 种不同的工具，可以在约 100 种图像格式之间转换。NetPBM 使用命令行操作，没有图形界面。

大部分的 Linux 发行版和 Cygwin 项目都包含 NetPBM。Windows、MacOSX 和其他操作系统上的 NetPBM 程序可以从下面的网址下载

http://netpbm.sourceforge.net/

Irfanview Irfanview 是 Windows 上一款出色的图像软件，具有易于安装、简洁、快速等特点。Irfanview 支持各种文件格式的浏览和转换，并提供基本的图片编辑功能，包括裁剪、重采样、颜色和亮度的调整等。Irfanview 同时支持 GUI 图形界面和命令行操作，包括批处理模式。

例如，当 Irfanview 的可执行程序 i_view32.exe 添加到 Windows 的环境变量后，如下命令

```
i_view32 *.gif /convert=*.png
```

会为当前目录下所有的 gif 文件创建 png 版本，保存的文件名为原 gif 文件的文件名加上 .png 后缀。Irfanview 可以从下面的网址下载

http://www.irfanview.com/

Irfanview 对于个人、学术和非盈利用户是免费的，商业用户则要求注册费。

Graphic Converter Graphic Converter 是一个 Macintosh 上的共享软件，可以读取约 190 种图片格式，并输出约 75 种格式。更多信息可见

http://www.lemkesoft.de/

WMF2EPS WMF2EPS 是运行在 Windows 上的共享软件，实现可以 `wmf` 到 `eps` 的转换。获取网址为

CTAN/support/wmf2eps/

KVEC KVEC 是一款自由软件，能将 `bmp`、`gif`、`tiff` 等位图格式转化为 PostScript 和其它的矢量图格式。KVEC 可以运行在 Windows、OS/2、Linux、Unix、Macintosh 和 BeOS 等系统上。更多信息可见

`http://www.kvec.de`

xv xv 是一个交互式图片操作程序，运行在 X Window 系统下。尽管有图像转换的功能，但是 xv 的设计目标是图片操作程序，因此不适用于图像转换。例如，它没有提供命令行功能，所以图像转换必须一个一个做。

xv 对于非商业使用是共享软件，对于商业使用则要求注册。更多信息可见

`http://www.trilon.com/xv/xv.html`

GIMP GIMP，即 GNU Image Manipulation Program，是一个图片操作的自由软件。它基本上提供了 PhotoShop 的功能。GIMP 可以运行在 Unix/Linux、Windows 和 MacOS 上。更多信息可见

`http://www.gimp.org/`

6.3 Level 2 EPS 封装

Level 2 PostScript 支持若干种压缩格式，包括用于 `jpeg` 文件的 DCT 和用于许多 `tiff` 文件的 LZW。此外，二进制数据可以是 ASCII 编码的，比如 ASCII85 或者 ASCIIHex，这样生成的 ASCII 文件大小分别是原来二进制文件的 125% 和 200%。由于 Level-2 `eps` 支持这些压缩格式，因此可以创建为 `jpeg` 或是 `tiff` 文件的封装。这样生成的 `eps` 比起传统的 `eps` 转换来说具有更高的质量和更小的体积。如果你有一台 Level 2 PostScript 打印机，那么你最好是用下面的这些封装程序来替代上节中的那些转换软件。不过由于这样得到的 PostScript 文件只能在 Level 2 PostScript 打印机上打印，会降低文件的通用性。

请注意，默认情况下，dvips 会剥离使用的 `eps` 图像的注释行（即以 `%%` 开头的行）。由于 ASCII85 编码的 Level 2 图像会包含以 `%%` 开头的行，所以，使用 ASCII85 编码的 level-2 `eps` 文件的用户必须使用`dvips -K0` 选项（`K` 加上数字 `0`），以阻止 dvips 剥离注释行。不过，ASCIIHex 编码的 level-2 文件则不会有这个问题。

jpeg2ps jpeg2ps 是一个用 C 语言编写的程序，可将 `jpeg` 图像转换为 Level 2 PostScript。jpeg2ps 可在 Unix、DOS 和其它操作系统下使用。jpeg2ps 可以从如下网址获得

CTAN/support/jpeg2ps/
`http://gnuwin32.sourceforge.net/packages/jpeg2ps.htm`

jpeg2ps 支持三种 level-2 编码：ASCII85 （默认值）、8-bit binary （使用 `jpeg2ps -b` 选项）、7-bit ASCIIHex （使用 `jpeg2ps -h` 选项）。

tiff2ps tiff2ps 可以运行在 Unix、DOS、Mac 和 VMS 系统上。`tiff` 图像可用 tiff2ps 程序转换为 LZW–编码的 Level 2 PostScript。`tiff2ps` 的源代码可以从如下站点获得

```
http://www-mipl.jpl.nasa.gov/~ndr/tiff/html/tools.html
ftp://ftp.sgi.com/graphics/tiff/
```

ImageMagick 的 level-2 EPS 功能 正如第 18 页的 6.2 节所述，ImageMagick 可以在一大类图像格式之间转换。其中一种格式就是 level-2 `eps`，因此，ImageMagick 具有和上述 level-2 封装相同的功能。例如，

```
convert file.jpeg file.eps2
```

会为 `file.jpeg` 创建一个 level-2 `eps` 版本 `file.eps2`

6.4 编辑 PostScript

虽然可直接编辑 `eps` 文件中的 PostScript 命令来改变图像，但这对大多数人来说还是很困难的。所幸的是，借助于下面的一些工具软件，可以很容易地编辑 `eps` 图像。

pstoedit pstoedit 是 Unix、Windows、DOS 和 OS/2 下的自由软件，它能够将 PostScript 或 PDF 图像转为其它矢量格式（比如 `Xfig` 的 `.fig` 格式）。更多信息可参考

```
http://www.pstoedit.com/
```

Mayura Draw Mayura Draw 以前称为 PageDraw，是 Windows3.1/95/NT 下的共享软件。当与 ghostscript 一起使用时，它可以编辑 PostScript 文件。见：

```
http://www.mayura.com/
```

xfig xfig 是 Unix/Xwindows 下功能强大的自由绘图软件，能够导入 `eps` 图形并加上标记。不过目前还不能改变原始的 `eps` 图像。参见

```
http://www.xfig.org/
```

第二部分　LaTeX 图形宏集

这一部分简要介绍 LaTeX 图形宏集。更多细节可以参考图形宏集文档 [1] 或者 *LaTeX Graphics Companion*[4]。

7　图像文件加载

导入图像需要使用 graphicx 宏包的 \includegraphics 命令，语法为

```
\includegraphics[选项]{文件}
```

这里选项在表 1、2、3 中列出。因为 \includegraphics 不会结束当前段落，所以它能够在文本中放置图形如  和  。

7.1　图形驱动

用户必须指定一个图形驱动以告诉图形宏包如何处理导入的图像。目前 LaTeX 图形宏集支持 18 种不同的驱动 [14]，不过，本文档只涉及几种最常用的驱动：dvips、dvipdfmx、pdftex、xetex、luatex。除了 dvipdfmx 之外，余下几种驱动一般不用显式指明，因为绝大部分 LaTeX 发行版中的 graphics.cfg 可以自动识别[15]。

指定驱动

如果用户想要指定一个驱动，可以用以下三种方式

1. 默认值在 graphics.cfg 文件中指定。
2. 任何通过 \documentclass 的选项指定的驱动会覆盖由 graphics.cfg 文件指定的驱动。
3. 任何由 \usepackage{graphics} 的选项指定的驱动会覆盖前两种方式指定的驱动。

7.2　DVIPS 模式下图像加载

dvips 模式下支持最好的图像格式是 eps。当使用 latex 编译文件时，下面的命令

```
\includegraphics{file.eps}
```

会从 file.eps 文件中按照自然大小导入图像。如果文件名没有扩展名，

```
\includegraphics{file}
```

那么 \includegraphics 会按照 \DeclareGraphicsExtensions 扩展名列表中自动加上扩展名（见 32 的第 9.1 节）。

7.3　pdfLaTeX 模式下的图像加载

pdfTeX 支持直接导入 pdf、png、jpeg 和 METAPOST 图像。当使用 pdflatex 编译时，下面的命令

[14] 到目前（2016 年）为止，支持的驱动数已经超过了 20 种。——译注

[15] graphics.cfg 文件会检测文件的编译方式。对于 latex 会指定 dvips 选项，对于 pdflatex 会指定 pdftex 选项，对于 xelatex 会指定 xetex 选项，对于 lualatex 会指定 luatex 选项。而对于 latex+dvipdfmx 编译方式，则需要手动声明 dvipdfmx 选项，因为经由 latex 生成 dvi 文件后，graphics.cfg 会假定接下来使用 dvips。

```
\includegraphics{file.pdf}
\includegraphics{file.png}
\includegraphics{file.jpg}
\includegraphics{file.mps}
```

会按照自然尺寸大小导入 pdf 文件 file.pdf、png 文件 file.png、jpeg 文件 file.jpg 以及 METAPOST 文件 file.mps。如果文件名没有扩展名

```
\includegraphics{file.eps}
```

那么 \includegraphics 会按照 \DeclareGraphicsExtensions 扩展名列表中自动加上扩展名（见 32 的第 9.1 节）。

7.4 同时供 LaTeX 和 pdfLaTeX 编译的文档

很多情况下要求文档同时可以由 LaTeX 或 pdfLaTeX 编译。要求 PostScript 输出时使用 LaTeX 编译和 dvips 驱动，要求 pdf 输出时则使用 pdfLaTeX 编译。在这两种方式之间切换会改变两件事情。[16]

- 合适的 graphicx 驱动会改变。
- 可以直接导入的图像类型会改变。

使用如下步骤会调整这些，这样一份文档可以同时由 LaTeX 和 pdfLaTeX 编译。

1. 对于每一份要导入的图像都创建两个副本[17]：
   (a) 一份 eps 版本，当使用 latex 编译时会导入。
   (b) 一份 png、pdf、jpeg 或者 METAPOST 版本，当使用 pdflatex 编译时会导入[18]。
2. 不要在 \documentclass 或者 \usepackage{graphicx} 命令中指定 dvips 或者 pdftex 选项。graphic.cfg 文件会自动传递合适的选项给 graphicx 宏包。
3. 当使用 \includegraphics 命令插图时，不要指定扩展名。例如

   ```
   \includegraphics{graphic}
   ```

   定义在 dvips.def 中的默认扩展名列表会使得 LaTeX 导入图像 eps 版本，而定义在 pdftex.def 中的默认扩展名列表会使得 pdfLaTeX 导入图像的 png、pdf、jpeg 或者 METAPOST 版本（见 32 的第 9.1 节）。
4. 不要直接使用 psfrag。如果需要进行 psfrag 替换，使用 53 页的第 15.1.5 节介绍的方法。

[16] 使用 xelatex 或者 lualatex 编译时无需考虑此问题。这是因为这两种编译方式能同时支持 eps 和 pdf、jpg、png 等图片格式，同时编译时也能够自动调用相应的 graphicx 驱动选项。——译注
[17] 有时使用 PurifyEPS （见 17 页的第 5.5 节）可以创建一个文件并可以同时供 LaTeX 和 pdfLaTeX 使用。
[18] 也可以使用 epstopdf 进行实时转换，见下面的第 7.4.2 节。——译注

7.4.1 使用 ifpdf 宏包的条件代码

ifpdf 宏包的 \ifpdf 命令会检测文档是否由 pdflatex 编译[19]，进而可以在文档中使用条件语句。

例如，为了精简扩展名列表（见第 9.1 节），可以用 \ifpdf 命令来定制

```
\usepackage{ifpdf}
...
\ifpdf
\DeclareGraphicsExtensions{.pdf,.png,.jpg,.mps}
\else
\DeclareGraphicsExtensions{.eps}
\fi
```

如果用户想要基于条件代码来使用不同的 \documentclass 选项，使用如下代码可以在 \documentclass 之前定义 \ifpdf

```
\RequirePackage{ifpdf}
\ifpdf
\documentclass[pdftex]{article}
\else
\documentclass[dvips]{article}
\fi
```

这段代码当使用 pdflatex 时会传递 pdftex 选项，而使用 latex 编译时会传递 dvips 选项。不过，正如 22 页的第 7.1 节所述，这段代码实际上没有必要，因为绝大部分的发行版本会根据 graphics.cfg 文件自动处理驱动选项。

7.4.2 使用 epstopdf 宏包

如果已经准备了 eps 图像而想使用 pdflatex 编译方式，可以考虑使用宏包 epstopdf。该宏包可以在编译时实时地将 eps 转成 pdf 格式图像，从而可以在 pdflatex 编译方式下使用。[20]

使用该宏包要注意两点：

- 需要开启 \write18 特性，从而使得可以在编译时调用外部程序。具体来说，编译方式为

  ```
  pdflatex -shell-escape test.tex (TeXLive 中)
  pdflatex -enable-write18 test.tex (MiKTeX 中)
  ```

- 在图形宏包 graphics/graphicx 之后载入 epstopdf，即

  ```
  \usepackage{graphicx}
  \usepackage{epstopdf}
  ```

结合之前介绍的 ifpdf，可以使用如下代码

[19] 历史上，有一种检测方法是基于 \pdfoutput 仅在使用 pdfLaTeX 时才有定义这一事实。然而，现在绝大部分 TeX 发行版中的 latex 命令实际上是在 dvi 模式下执行 pdfLaTeX。此时无论用 latex 和 pdflatex 中的哪种来编译，\pdfoutput 都有定义。ifpdf 宏包则提供了一个鲁棒的条件命令，可以判断文档是否直接处理成 pdf 文件，从而解决了这一问题。

[20] 在 TeXLive（2010 之后的较新版本）中会自动调用该宏包，也就是说，此时可以直接用 pdflatex 编译 eps 插图的情况——译注

```
\usepackage{graphicx}
\usepackage{ifpdf}
\ifpdf
\usepackage{epstopdf}
\fi
```

这样，在只准备 eps 格式图像的情况下，dvips 和 pdflatex 都可以处理文档。

7.5 指定宽度、高度和角度

指定宽度

如下命令

```
\includegraphics[width=3in]{file}
```

导入指定的图片文件，且宽度为 3 英寸[21]。不过，为了使图像的页面布局更具通用性，图片宽度最好指定为可伸缩的相对长度而不是用像 3 英寸这样的固定尺寸[22]。例如，

- 如下命令将插入的图片伸缩至和当前一行文本的宽度相同：

  ```
  \includegraphics[width=\linewidth]{graphic}
  ```

- 如下命令使得插入图片的宽度为当前一行文本的 80%：

  ```
  \includegraphics[width=0.80\linewidth]{graphic}
  ```

- 当与 calc 宏包配合使用时，如下命令可以使得插入图片的宽度比当前一行文本的宽度窄 2 英寸：

  ```
  \includegraphics[width=\linewidth-2.0in]{graphic}
  ```

指定高度

类似地，如下命令

```
\includegraphics[height=2cm]{file}
```

导入指定的图片文件，且使得高度为 2 厘米。此外，\includegraphics 命令还有一个 totalheight 选项来指定整体高度（关于高度和整体高度的定义可参见第 11 页的第 2 节）。

指定角度

\includegraphics 命令的 angle 选项可以指定插入图像的角度：

```
\includegraphics[angle=45]{graphic}
```

该命令会按照原始尺寸插入图片，并且按逆时针旋转 45 度。

7.5.1 同时指定角度和高度/宽度

由于 \includegraphics 的选项是从左到右依次解释的，所以其中指定角度和大小顺序会导致不同的效果。例如

[21] 同时高度也会按相应的比例缩放——译注

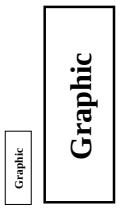
[22] 预先定义的相对长度包括：
\textwidth 是文档中正文的宽度；
\linewidth 是当前环境中一行的宽度；
em 是当前字体大写字母 M 的宽度；
ex 是当前字体小写字母 x 的高度。

表 1: \includegraphics 选项

height	图形的高度（可为任何 TEX 度量单位）。
totalheight	图形的整体高度，可为任何 TEX 度量单位。
width	图形的宽度（可为任何 TEX 度量单位）。
scale	图形的缩放因子，设定 scale=2 会使插入图形的大小为其自然大小的两倍。
angle	设定旋转的角度，以度为单位，逆时针方向为正。
origin	origin 指定图形绕那一点旋转，默认是围绕参考点旋转。origin 点可以与第 8.3节的 \rotatebox 命令中的一样。比如 origin=c 将使图形绕它的中心旋转。
bb	设定 BoundingBox 的值。例如，bb=10 20 100 200 会设定 BoundingBox 的左下角在 (10, 20)，右上角在 (100, 200)。因为 \includegraphics 会自动从 eps 文件中读入 BoundingBox 值，所以一般不用 bb 这个选项。但它在 eps 文件中的 BoundingBox 丢失或不准确时还是很有用的。

```
\begin{center}
\includegraphics[angle=90,totalheight=1cm]{graphic}
\includegraphics[totalheight=1cm,angle=90]{graphic}
\end{center}
```

会生成如下结果



第一个盒子是先旋转了 90 度，然后缩放使得整体高度为 1 厘米。第二个盒子先缩放使得整体高度为 1 厘米，然后再旋转 90 度。另外要注意的是，以上图形中两幅图像之间有一个空格，因为第一行 \includegraphics 的末尾并没有以 % 结束。

7.6 一些例子

下面是一些使用 \includegraphics 命令来插入图形的例子。[23] 这里为方便起见，定义水平线 \HR 为

```
\newcommand{\HR}{\rule{1em}{0.4pt}}
```

以下几个例子中使用了 scale、width、height、angle 以及 keepaspectratio 等选项及其不同的顺序，从而得到的不同效果。

[23] 本小节来自于王磊在旧版本的中译本《LATEX 2ε 插图指南》中额外补充的内容。——译注

表 2: \includegraphics 裁剪选项

viewpoint	指定图像可以被看到的部分。如同 BoundingBox 一样，这个区域由四个数字决定，分别是左下角和右上角的坐标。这里的坐标是以 BoundingBox 左下角为起点的相对坐标。 例如，如果图像的 BoundingBox 的值是 50 50 410 302，那么 viewpoint=50 50 122 122 显示的图像是一平方英寸大小的正方形区域，且以图像左下角为左下角。而 viewpoint=338 230 410 302 显示的图像也是一平方英寸大小的正方形区域，但以图形的右上角为右上角。 必须使用 clip 选项（见表 3）来阻止显示视图以外的部分。
trim	显示部分图像的另一种方法。所给出的四个数字分别代表了从左、下、右、上被截去的值。正数代表从此方向截去的大小，而负数则代表从此方向加上的大小。 例如，trim=1 2 3 4 代表图像的左边截去 1 bp，下边截去 2 bp，右边截去 3 bp，上边截去 4 bp。 必须使用 clip 选项（见表 3）来阻止显示被截去的部分。

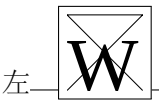
表 3: \includegraphics 布尔型选项

clip	指定 clip=false 会显示整个的图形，即使有些部分在视图之外。（默认值） 指定 clip=true 时将不显示图形在视图之外的部分。
draft	使用 draft 或 draft=true 选项将阻止图像的导入。此时在插图的位置只显示图像的 BoundingBox 和文件名，这会加快文档的显示和打印速度。使用 draft=false 则会显示图像。 如果使用 draft 宏包选项 \usepackage[draft]{graphicx}，那么文档中的所有图像都以 draft 方式插入。
keepaspectratio	在没有设定 keepaspectratio 选项时，如果同时给定图像的宽度和高度/整体高度，那么为了满足所设定的高和宽，图像的缩放可能会失真变形。 在设定 keepaspectratio 选项后，给定图像的宽度和高度/整体高度时，图像会在保持原有的宽高比例下缩放，尽可能使得图像满足所设定的高和宽，但是图像的实际宽高不会超出设置的值。

左  右

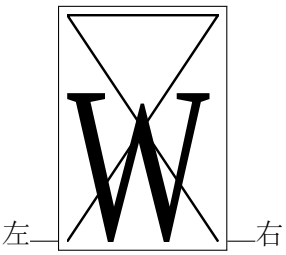
例一

```
左\HR\fbox{%
\includegraphics
[scale=.5]{w.eps}%
\HR 右
```

左  右

例二

```
左\HR\fbox{%
\includegraphics%
[width=10mm]{w.eps}%
\HR 右
```

左  右

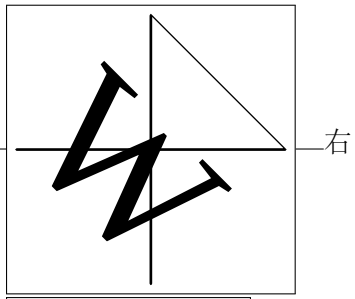
例三

```
左\HR\fbox{%
\includegraphics
[height=20mm,width=30mm]%
{w.eps}}\HR 右
```

左  右

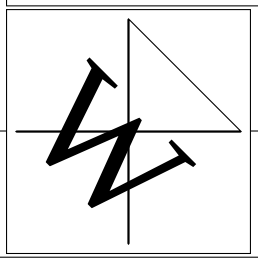
例四

```
左\HR\fbox{%
\includegraphics
[height=20mm,width=30mm,%
keepaspectratio]{w.eps}}%
\HR 右
```

左  右

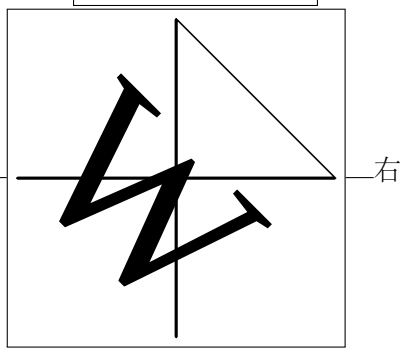
例五

```
左\HR\fbox{%
\includegraphics
[angle=-45]{w.eps}}%
\HR 右
```

左  右

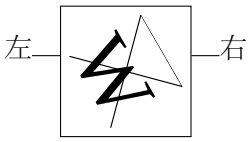
例六

```
左\HR\fbox{%
\includegraphics
[angle=-45,width=30mm]%
{w.eps}}\HR 右
```

左  右

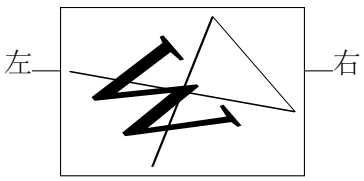
例七

```
左\HR\fbox{%
\includegraphics
[width=30mm,angle=-45]%
{w.eps}}\HR 右
```



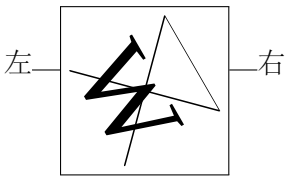
例八

```
左\HR\fbox{%
\includegraphics
[angle=-60,totalheight=15mm]%
{w.eps}}%
\HR 右
```



例九

```
左\HR\fbox{%
\includegraphics
[angle=-60,totalheight=20mm,%
width=30mm]{w.eps}}%
\HR 右
```



例十

```
左\HR\fbox{%
\includegraphics
[angle=-60,totalheight=20mm,%
width=30mm,keepaspectratio]%
{w.eps}}%
\HR 右
```

8 旋转和缩放对象

除了 \includegraphics命令外，graphicx 宏包还提供了另外四个命令用来旋转和缩放任意的 LaTeX 对象，例如文本、图像等。

```
\scalebox{水平缩放因子}[垂直缩放因子]{对象}
\resizebox{宽度}{高度}{对象}
\resizebox*{宽度}{整体高度}{对象}
\rotatebox[选项]{角度}{对象}
```

因为 graphicx 包的 \includegraphics 已经带有 angle、width 等支持旋转和缩放的选项，所以本节介绍的这几个命令很少在插图时使用。例如：

```
\includegraphics[scale=2]{file.eps}
\includegraphics[width=4in]{file.eps}
\includegraphics[angle=45]{file.eps}
```

上述命令和下面的命令等到的结果是相同的。

```
\scalebox{2}{\includegraphics{file.eps}}
\resizebox{4in}{!}{\includegraphics{file.eps}}
\rotatebox{45}{\includegraphics{file.eps}}
```

尽管结果相同，但最好还是用前一种方法，因为它的速度更快，而且生成 PostScript 和 pdf 的效率更高。

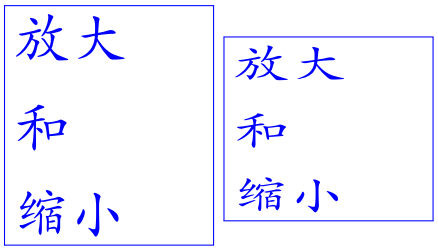
8.1 scalebox 命令

语法：

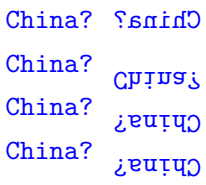
```
\scalebox{水平缩放因子}[垂直缩放因子]{对象}
```

\scalebox 命令对其作用的对象进行缩放，使得缩放后对象的宽度为原始宽度与水平缩放因子之积，高度为原始高度与垂直缩放因子之积。如果垂直缩放因子没有给出，那么默认取为水平缩放因子，以保持原始的宽高比进行缩放。如果缩放因子为负值，则对对象进行翻转。下面是几个例子


```
\scalebox{2}{这是放大的文字} \\
这是正常的文字  \\
\scalebox{.5}{这是缩小的文字}
```




```
\framebox{\scalebox{2}{%
\parbox{.5in}{放大 \\ 和 \\ 缩小}}}
\framebox{\scalebox{2}[1.5]{%
\parbox{.5in}{放大 \\ 和 \\ 缩小}}}
```




```
China? \scalebox{-1}[1]{China?} \\
China? \scalebox{1}[-1]{China?} \\
China? \scalebox{-1}[-1]{China?} \\
China? \scalebox{-1}{China?}}
```

8.2 resizebox 命令

语法：

```
\resizebox{宽度}{高度}{对象}
\resizebox*{宽度}{整体高度}{对象}
```

\resizebox 命令将对象的大小改变为给定值。如果宽度或高度中的任一项用 ! 给出，那么缩放时会保持原有的宽高比。例如：

```
\resizebox{2in}{!}{argument}
```

将对象的宽度改变为 2 英寸，同时保持宽高比。

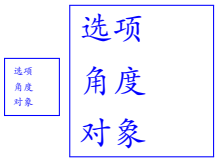
标准的 LaTeX 2ε 长度命令 \height、\width、\totalheight、\depth 可用来表示对象的原始尺寸。因此，

```
\resizebox{2in}{\height}{argument}
```

使得对象的宽度改变为 2 英寸但保持原有高度不变。

除了以外，命令 \reseizebox* 与 \resizebox 几乎是相同的不同之处仅在于第二个参数表示对象的全部高度（关于高度和整体高度的定义见第 2 节，关于 height 和 totalheight 选项的比较见第 11.1 节。）

下面是几个例子：




```
\framebox{\resizebox{5mm}{!}{%
\parbox{14mm}{选项 \\ 角度 \\ 对象}}}
\framebox{\resizebox{!}{10mm}{%
\parbox{14mm}{选项 \\ 角度 \\ 对象}}}
```


```
\resizebox*{2cm}{3cm}{\LaTeX{}~图形} \\
\resizebox*{2cm}{1cm}{\LaTeX{}~图形}
```

8.3 rotatebox 命令

语法：

```
\rotatebox[选项]{角度}{对象}
```

\rotatebox 将对象旋转一给定度数的角度，逆时针方向为正。默认选项下对象绕它的参考点旋转。\rotatebox 命令中选项允许对象绕给定的点来旋转。

1. 给定 [x=xdim,y=ydim]，则对象旋转所绕的点相对于参考点的坐标为 (xdim, ydim)。
2. origin 选项可以指定 12 个特殊点其中之一（见图 3）。

origin 点的水平位置由 lcr （分别代表左、中、右）三个字母其中之一确定，垂直位置则由 t,c,B,b（分别代表顶部、中部、基线、底部）四个字母中的一个来确定。例如：

[rb] 右下角。

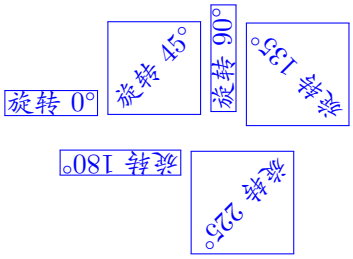
[lt] 左上角。

[cB] 图形基线的中点。

几点说明：

- 标记字母的顺序并不重要，[br] 等同于 [rb]。
- c 代表水平位置的中点还是垂直位置的中点由和它一起的标记字母来决定。
- 如果只给出一个标记字母，那么另一个将被假设为 c。即，[c] 等于 [cc]，[l] 等于 [lc]，[t] 等于 [ct]，等等。

下面是一个例子：




```
\setlength{\fboxsep}{0mm}
\newcommand{\MyRot}[1]{%
\fbox{\rotatebox{#1}{旋转~$#1^\circ$}}}
\MyRot{0} \MyRot{45} \MyRot{90}
\MyRot{135} \MyRot{180} \MyRot{225}
```

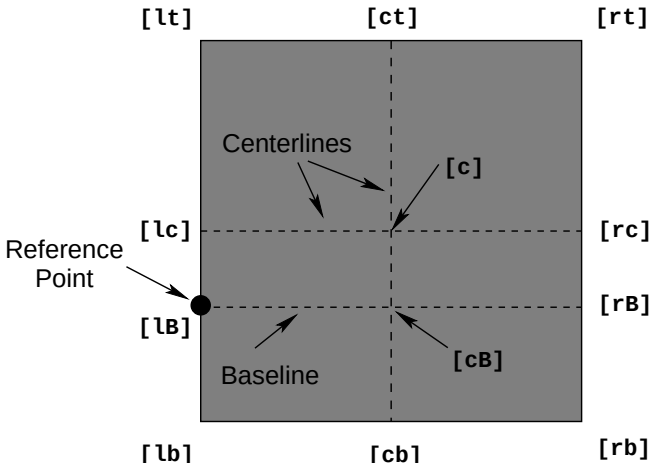


图 3: 可用的 origin 点

9 高级插图命令

本节描述了一些 LaTeX 2ε 图形宏包套件的高级命令，用于下述情形。

1. 当使用没有扩展名的文件名时。如：

   ```
   \includegraphics{file}
   ```

2. 当使用压缩的 eps 图形文件时。见第 14.1 节。

3. 当使用非 eps 格式的图形文件时。见第 14.2 节。

在这些情况下，LaTeX 为了处理由命令 \includegraphics 所导入的图像文件，就需要用 \DeclareGraphicsRule 和 \DeclareGraphicsExtensions 命令来控制。

- \DeclareGraphicsExtensions 命令指定了在没有提供图形文件扩展名的情况下，LaTeX 将自动为其加上的扩展名列表（如 .eps、.ps、.eps.gz 等）。
- \DeclareGraphicsRule 命令指定了对图形文件执行的命令。执行这一命令要求操作系统支持管道功能，比如 Unix 等操作系统，而 DOS 则不行。

  若将此命令指定为一解压缩命令，那么就可以使用压缩的 eps 图形文件。若将此命令指定为一图形格式转换命令，那么就可以使用非 eps 格式的图形文件。

9.1 DeclareGraphicsExtensions 命令

\DeclareGraphicsExtensions 命令告诉 LaTeX，若 \includegraphics 命令所引入的文件没有提供扩展名，将试图为其自动加上什么样的扩展名。

为方便起见，在选择图形驱动[24] 时，就已经有一个相应的预设扩展名集。举例来说，如果选择 dvips 作为图形驱动，那么默认会使用下列图形文件扩展名（在 dvips.def 中定义）：

```
\DeclareGraphicsExtensions{.eps,.ps,.eps.gz,.ps.gz,.eps.Z}
```

[24] 指定一个图形驱动选项如 \usepackage[dvips]{graphics} 将会覆盖掉在 graphics.cfg 中设定的缺省驱动选项——旧译本注。

如果 graphicx 宏包使用 pdftex 驱动，那么默认会使用下列图形文件扩展名（在 pdftex.def 中定义）[25]：

```
\DeclareGraphicsExtensions{.png,.pdf,.jpg,.mps}
```

在 dvips 图像扩展名列表中，\includegraphics{file} 首先寻找 file.eps，其次是 file.ps，再其次是 file.eps.gz，直到找到一个文件。相应地，可以用

```
\includegrapincs{file}
```

取代

```
\includegrapincs{file.eps}
```

这样做的好处是如果以后决定压缩 file.eps，也无须更改 LaTeX 文件。无后缀名的方式使得可以同时使用 LaTeX 和 pdfLaTeX 编译文档，见第 23 页的 7.4 节。然而，这种无后缀名的方式可能会加剧内存池空间问题，详见下面的 9.1.2 小节。

9.1.1 无扩展名的文件

需要注意的是，

```
\includegrapincs{file}
```

不会试图寻找 file，除非空扩展名列表中包含空的扩展名 {}。例如：

```
\DeclareGraphicsExtensions{.eps,.eps.gz,{}}
```

将试图在没找到 file.eps 和 file.eps.gz 的情况下寻找 file。

由于默认的扩展名列表中不包含空扩展名，如果想要使用无扩展名文件的话，必须使用 \DeclareGraphicsExtensions 命令定义一个包含空扩展名的列表。

9.1.2 池空间问题

不给出扩展名而靠 LaTeX 从 \DeclareGraphicsExtensions 的扩展名列表中选择正确的扩展名可能加剧池空间问题（见第 13.4 节）。如果有池空间问题的话，应当使用 \DeclareGraphicsExtensions 指定的扩展名数目尽可能小。如：

```
\DeclareGraphicsExtensions{.eps,.eps.gz}
```

9.2 DeclareGraphicsRule 命令

\DeclareGraphicsRule 命令指定 \includegraphics 如何按照扩展名来对操作图像文件。语法为

```
\DeclareGraphicsRule{ext}{type}{sizefile}{command}
```

例如，如下命令

[25] 实际上在较新版本（2016 年）中，dvips.def 中定义扩展名的方式为

```
\def\Gin@extions{.eps,.ps,.eps.gz,.ps.gz,.eps.Z,.mps}
```

pdftex.def 中定义扩展名的方式为

```
\def\Gin@extions{.png,.pdf,.jpg,.mps,.jpeg,.PNG,.PDF,.JPG,.JPEG}
```

——译注

表 4: \DeclareGraphicsRule 的选项

ext	文件的扩展名。
type	扩展名所对应的图像类型。
sizefile	包含图像 BoundingBox 信息的文件的扩展名。如果这一选项为空，那么必须要在 \includegraphics 命令中给定 bb 选项的值。
command	作用于图像文件的命令，此项常为空。命令前必须有一个前单引号（键盘上数字键 1 左侧的键——译注），注意不是常用的后单引号。目前为止，只有 dvips 能够执行这样的命令。参见第 14.2 节关于如何用这样的命令来处理非 eps 格式图像和压缩的 eps 图像的例子。

```
\DeclareGraphicsRule{.eps.gz}{eps}{.eps.bb}{`gunzip -c #1}
```

指定任何以 .eps.gz 为扩展名的文件为压缩的 eps 文件，该文件的 BoundingBox 信息存放在扩展名为 .eps.bb 的文件中，并用命令 gunzip -c 来解压缩（因为 LaTeX 不能从压缩文件中读取 BoundingBox 信息，所以 BoundingBox 行必须存放到非压缩文件中）。

可以允许有多个 \DeclareGraphicsRule 命令。

\DeclareGraphicsRule 命令允许使用 * 代表任何未知扩展名，例如：

```
\DeclareGraphicsRule{*}{eps}{*}{}
```

会导致所有未知扩展名的文件都被认为是 eps 文件，比方说 file.EPS 就被当做 eps 文件。

文件名中的句点

文件的扩展名定义为文件名里第一个句点以后的部分，这样做是为了可以将 .eps.gz 结尾的文件识别成压缩的 eps 文件。为了避免混淆，文件的基本名中不要使用句点。例如，指定文件 file.name.eps.gz 会让 \includegraphics 寻找扩展名为 .name.eps.gz 所对应的图像规则。由于这样的规则很有可能不存在，结果导致使用未知扩展名所对应的规则。例外的情形是该文件的格式正好是缺省格式，如果未知扩展名的文件都被认为是 eps 文件，那么 file.name.eps 就恰巧能被正确地识别。

预定义命令

为方便起见，根据不同的图形驱动选项，已经预定义了不同的缺省图像规则。例如使用 dvips 图形驱动选项时，文件 dvips.def 定义了如下缺省图形规则[26]：

```
\DeclareGraphicsRule{.eps}{eps}{.eps}{}
\DeclareGraphicsRule{.ps}{eps}{.ps}{}
\DeclareGraphicsRule{.pz}{eps}{.bb}{`gunzip -c #1}
\DeclareGraphicsRule{.eps.Z}{eps}{.eps.bb}{`gunzip -c #1}
\DeclareGraphicsRule{.ps.Z}{eps}{.ps.bb}{`gunzip -c #1}
\DeclareGraphicsRule{.eps.gz}{eps}{.eps.bb}{`gunzip -c #1}
\DeclareGraphicsRule{.ps.gz}{eps}{.ps.bb}{`gunzip -c #1}
\DeclareGraphicsRule{.pcx}{bmp}{}{}
\DeclareGraphicsRule{.bmp}{bmp}{}{}
\DeclareGraphicsRule{.msp}{bmp}{}{}
\DeclareGraphicsRule{*}{eps}{*}{}
```

[26] 实际上 dvips.def 的代码并没有使用 \DeclareGraphicsRule，不过效果是一样的。

前面两个命令定义扩展名为 .eps 和 .ps 的文件为 eps 文件，它们后面的五个命令定义了压缩 eps 文件的扩展名和解压命令，接下来的三个命令定义了位图文件 pcx、bmp、msp 的扩展名，最后一个命令设定未知扩展名的文件按照 eps 文件处理。

如果使用 pdfTEX引擎，那么 pdftex.def 缺省定义了以下的图形规则[27]：

```
\DeclareGraphicsRule{.png}{png}{.png}{}
\DeclareGraphicsRule{.pdf}{pdf}{.pdf}{}
\DeclareGraphicsRule{.jpg}{jpg}{.jpg}{}
\DeclareGraphicsRule{.mps}{mps}{.mps}{}
```

这样就指定了 pdfTEX 支持的图像格式的处理方式。

[27] pdftex.def 实际上先检查 pdfTEX 的版本，然后根据不同版本的处理方式定义相应的图形规则。

第三部分　LaTeX 图形命令的使用

10　水平间距和居中

10.1　水平居中

图形的放置位置由当前文本的排列方式所决定。为使图形居中放置，可将其放入居中环境 center 中。

```
\begin{center}
\includegraphics[width=2in]{graphic.eps}
\end{center}
```

如果 \includegraphics 命令处于一个环境中（例如 minipage 或 figure），用 \centering 可将其后的内容居中排列。例如：

```
\begin{figure}
\centering
\includegraphics[width=2in]{graphic.eps}
\end{figure}
```

效果类似于

```
\begin{figure}
\begin{center}
\includegraphics[width=2in]{graphic.eps}
\end{center}
\end{figure}
```

这里推荐使用 \centering，因为 \begin{center} 会使图形上下方的垂直间距增加一倍（figure 带有的垂直间距加上 center 环境带有的垂直间距）。若希望有额外的垂直间距，可使用第 19.1 节介绍的命令。

过时的用法

\psfig 和 \epsfbox 命令的缺陷让它们很难使图形居中排列。过去，作为一种解决办法，使用了 TeX 命令 \centerline 和 \leavevmode。而 \includegraphics 命令已克服了这些缺陷，允许直接与 \centering 命令一起使用或用在 center 环境中，因此也就不需再使 \centerline 和 \leavevmode了。

10.2　水平间距

LaTeX 排列图形的方式实际上与排列其它对象（比如文字）是一样的，了解到这一点很重要。举例来说，如果行尾不是以 % 结束的话，LaTeX 会自动在两行之间加进一个字符的水平间距。例如，

```
Hello
World
```

在输出结果中 “Hello” 和 “World” 之间会有一个字符的水平间距。类似地，

```
\includegraphics{file}
\includegraphics{file}
```

则在图形之间有一个字符的水平间距。在第一行的行尾加上注释符%

```
\includegraphics{file}%
\includegraphics{file}
```

就会取消图形之间的水平间距。

如果需要水平间距，可用 \hspace 命令在图形之间加入指定长度[28] 或用 \hfill 加入一个可填充可能的间距的弹性长度。例如：

```
\includegraphics{file.eps}\hfill\includegraphics{file.eps}
```

将两个图形尽量向左右分开。而

```
\hspace*{\fill}\includegraphics{file.eps}%
\hfill\includegraphics{file.eps}\hspace*{\fill}
```

使得图形的两边和中间的间距都相等。由于断行前的 \hfill 命令会被忽略，所以需要用\hspace*{\fill} 来替代它。

其它间距命令

除了 \hspace 和 \hfill 之外，\quad 命令会插入当前字体大小的空白。例如，如果使用 10 pt 字体大小，那么 \quad 会插入 10 pt 的水平间距。而命令 \qquad 会插入两倍于 \quad 的水平间距。

11 旋转、缩放和对齐

11.1 高度和整体高度的区别

使用 height 选项时必须要小心，因为用户经常想要的其实是由 totalheight 选项设置的整体高度（参见 12 页的图 1）。当对象的深度为零时，对象的整体高度就是它的高度，此时使用 height 选项不会有什么问题。但是，当对象的深度不为零时，使用 height 而不是 totalheight 会导致不正确的图像大小或除零的错误。导入图像时，区分 height 和 totalheight 对于旋转和缩放时显得尤其重要。例如：

```
\includegraphics[angle=-45,totalheight=1in]{file}
\includegraphics[angle=-45,height=1in]{file}
```

第一个命令缩放一个旋转了的图形，使其全部高度为 1 英寸。而第二个命令缩放一个旋转了的图形，使其在参考点以上的部分为 1 英寸高。

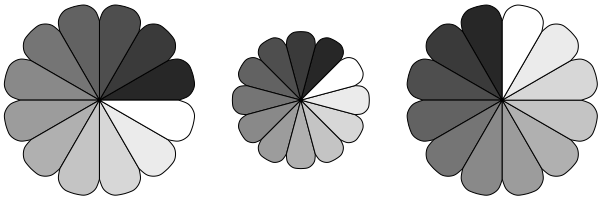
11.2 旋转图形的缩放

当在插图命令中指定高度或宽度时，这里给出的大小并不是图形的大小，而是图形的 BoundingBox 的大小。这点在图形旋转和缩放时很重要。例如：

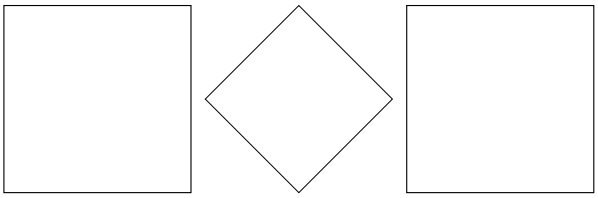
```
\begin{center}
\includegraphics[totalheight=1in]{rosette}
\includegraphics[angle=45,totalheight=1in]{rosette}
\includegraphics[angle=90,totalheight=1in]{rosette}
\end{center}
```

得到

[28] 为提高文档的通用性，可以使用 \textwidth 或 em （原文误作 \em——译注）等作为 \hspace 的参数，而不是采用固定长度。



尽管看上去图形的大小不一有点奇怪，但在看过它们的 BoundingBox 后就会明白了。



每一幅图像的结果是旋转后的 BoundingBox 为 1 英寸高。缩放功能改变的是 BoundingBox 的大小，而不是实际看到图像的大小。

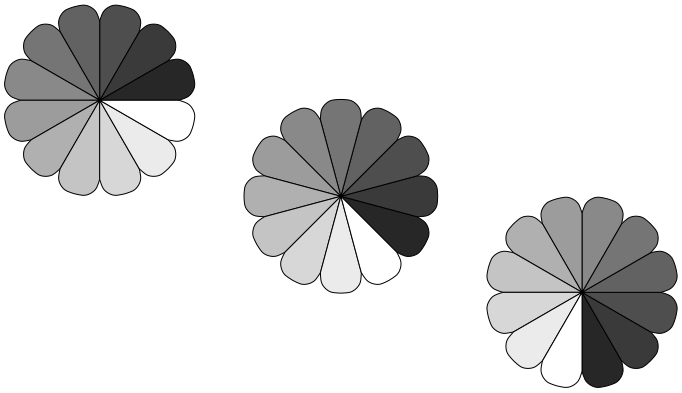
11.3 旋转图形的对齐

11.3.1 第一个例子

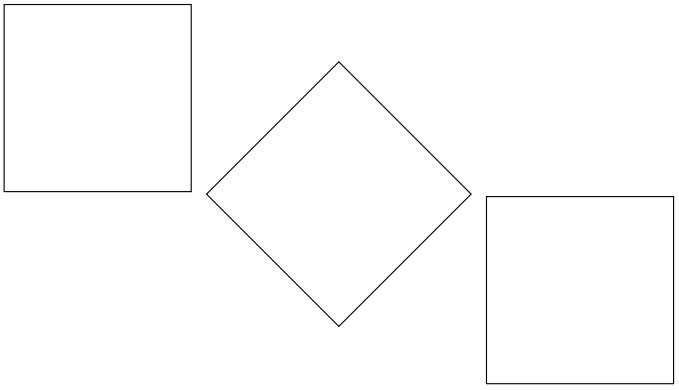
当图形被旋转时，可能会出现不对齐的情况。例如：

```
\begin{center}
   \includegraphics[totalheight=1in]{rosette}
   \includegraphics[totalheight=1in,angle=-45]{rosette}
   \includegraphics[totalheight=1in,angle=-90]{rosette}
\end{center}
```

得到



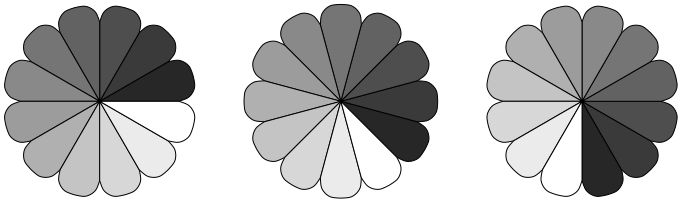
当然，这种现象仍可用图形的 BoundingBox 来解释。



在这种情况下，我们可以看到图形对象的参考点（左下角）是处于同一水平线上的。如果希望是中间对齐，那么可以用 \includegraphics 的 origin 选项。

```
\begin{center}
   \includegraphics[totalheight=1in]{rosette}
   \includegraphics[totalheight=1in,origin=c,angle=-45]{rosette}
   \includegraphics[totalheight=1in,origin=c,angle=-90]{rosette}
\end{center}
```

这次所有图形都是中间对齐的。

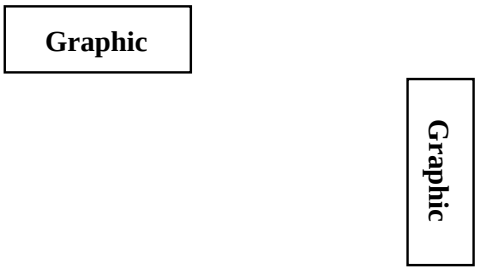


11.3.2 第二个例子

同样地，下面的命令

```
\begin{center}
\includegraphics[width=1in]{graphic}
\hspace{1in}
\includegraphics[width=1in,angle=-90]{graphic}
\end{center}
```

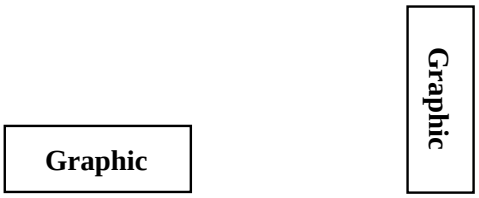
将右边的图形绕它的左下角旋转，得到如下结果：



要想使图形的底部对齐，使用下面的命令：

```
\begin{center}
\includegraphics[width=1in]{graphic}
\hspace{1in}
\includegraphics[width=1in,origin=br,angle=-90]{graphic}
\end{center}
```

上述命令让右边的图形绕它的右下角旋转，得到如下结果：



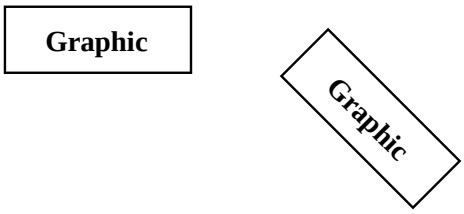


图 4: 带 [b] or [t] 选项的 minipage 环境

11.4 小页环境的垂直对齐

将图形放置于 minipage 小页环境中是经常遇到的情况下，而且也十分实用（见第 28 节）。当小页并列时，LaTeX 会将它们的参考点垂直对齐地排列。缺省地，小页的参考点是它的左边界的中点。可用一个可选参数项来改变小页的参考点的位置。

[b] 使小页的参考点与小页底行的参考点对齐。

[t] 使小页的参考点与小页顶行的参考点对齐。

注意选项 [b] 不会将参考点置于小页的底部（除非其底行的参考点在它的底部），同样地，选项 [t] 不会将参考点置于小页的顶部（除非其顶行的参考点在它的顶部）。

当小页中只有一行时，[b] 和 [t] 选项得到的结果是一样的。例如：

```
\begin{center}
\begin{minipage}[b]{.25\linewidth}
\centering
\includegraphics[width=1in]{graphic}
\end{minipage}%
\begin{minipage}[b]{.25\linewidth}
\centering
\includegraphics[width=1in,angle=-45]{graphic}
\end{minipage}
\end{center}
```

和

```
\begin{center}
\begin{minipage}[t]{.25\linewidth}
\centering
\includegraphics[width=1in]{graphic}
\end{minipage}%
\begin{minipage}[t]{.25\linewidth}
\centering
\includegraphics[width=1in,angle=-45]{graphic}
\end{minipage}
\end{center}
```

都得到图 4 的结果。在这两种情况下，小页的参考点都是图形的参考点（左下角）。

11.4.1 小页的底部对齐

让小页的底部对齐的方法之一是强制使小页的底部为其基线。再次要注意的是，小页环境 minipage 的 [b] 选项效果是让小页最底行的基线作为小页的基线。

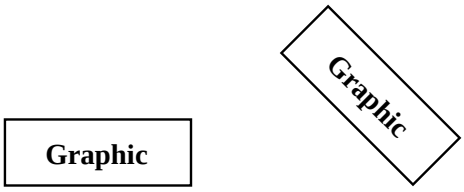


图 5: 底端对齐的小页环境

如果小页最底行正好是一条高和深都为零的线，那么最后一行的参考点就是该行的底部，这样 [b] 选项就可使其作为基线。类似地，如果在 \end{minipage} 之前添加一条高度和深度为零的线，那么 [b] 选项就会使小页的底部作为小页的基线。命令 \par\vspace{0pt} 就可以产生这样高和深都为零的线段。这时这条深度为零的线的基线就是小页的底部，选项 [b] 可以让小页的底部对齐了。例如：

```
\begin{center}
\begin{minipage}[b]{.25\linewidth}
\centering
\includegraphics[width=1in]{graphic}
\par\vspace{0pt}
\end{minipage}%
\begin{minipage}[b]{.25\linewidth}
\centering
\includegraphics[width=1in,angle=-45]{graphic}
\par\vspace{0pt}
\end{minipage}
\end{center}
```

结果如图 5。

11.4.2 小页的顶部对齐

当在小页的顶行加入一条高度和深度都为零的线段时，使用 [t] 选项使得小页的基线为它的顶部。如果在并列的若干小页环境中都进行这样的操作，那么就可以使这些小页环境的顶端对齐。

命令 \vspace{0pt} 可以在小页的顶端插入一条高度和深度都为零的线段。由于该条高度为零的线段的基线就位于小页的顶部，现在使用 [t] 选项就可以使得小页的顶部对齐了。例如：

```
\begin{center}
\begin{minipage}[t]{.25\linewidth}
\vspace{0pt}
\centering
\includegraphics[width=1in]{graphic}
\end{minipage}%
\begin{minipage}[t]{.25\linewidth}
\vspace{0pt}
\centering
\includegraphics[width=1in,angle=-45]{graphic}
\end{minipage}
```

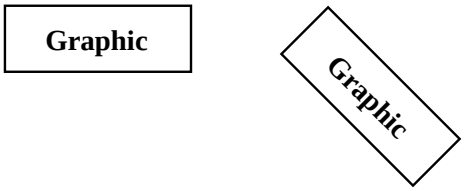


图 6: 顶部对齐的小页环境

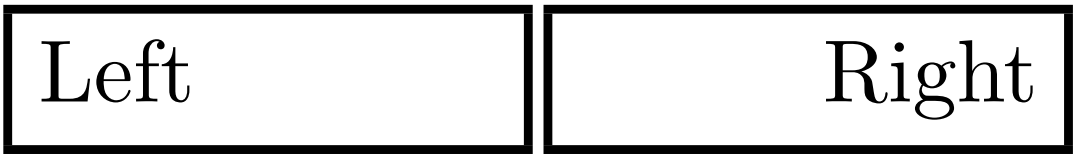


图 7: 两幅图像的内容

```
\end{center}
```

结果如图 6 所示。

这里小页的顶部是和当前基线对齐。如果要求小页的顶部是和当前文本行的顶部对齐，可用 `\vspace{-\baselineskip}` 代替 `\vspace{0pt}`。相关专题参考 [10, 第 863–865 页]。

12 两幅图像的堆叠

本节描述如何堆叠两幅图像。请注意这里没有检查确保顶层的图像是透明的。如果顶层图像创建时使用非透明的背景，那么就会隐藏底层图像。

例如[29]，文件 `left.eps` 和 `right.eps` 包含的图像如图 7 所示。使用如下命令

```
\makebox[0pt][l]{\includegraphics{left.eps}}%
\includegraphics{right.eps}
```

就会堆叠两幅图像，如图 8 所示。堆叠两幅图像时，它们参考点（左下角）是重合的。在这个例子中，两幅图像的自然大小是相同的，所以不需要缩放就可以完全堆叠在一起。其它的图像可能需要缩放（使用 `\includegraphics`、`\scalebox` 或是 `\resizebox` 等命令）才能达到想要的堆叠效果。

如果没能理解 `\makebox` 命令，这样的堆叠代码看起来就会显得有些不可思议。实际上，`\makebox[0pt][l]{...}` 命令会创建一个宽度为零的盒子。当指定宽度时（这里是 `0 pt`），那么排版算法就会分配这样宽度的水平空间，而不管其中内容的实际宽度是怎样。这样，对于一个内容居左的宽度为零的盒子，之后的 LaTeX 对象的排版效果就是覆盖在该盒子之上。

[29] 尽管在这个例子中使用了两个 `eps` 图像，类似的代码也可以用于堆叠其它图像格式。

图 8: 两幅堆叠的图像

13 使用子目录

当需要大量的图形文件时，你可能希望将它们存放到一个子目录下。例如，假设子目录的名字叫 sub，这时你试图用如下的命令来插入图形 file.eps。

```
\includegraphics{sub/file.eps}
```

尽管这种用法在大多数 Unix 和 DOS 下的 TEX 发行版里工作正常，它却有以下的问题：[30]

效率不高 每当 TEX 打开一个文件，该文件名就被存入 TEX 的内存中。当打开大量的文件时，内存空间减少会导致内存池大小的错误（见第 13.4 节）。显式地给出子目录名增加了文件名的长度，进而加重这种池空间问题。

通用性差 LATEX 的一大优势就是它的文件能在任何操作系统平台上使用。然而，在文件名中包括子目录名会使文件依赖于操作系统。如果不作明显的改变，上面的例子就无法在 VMS 或 Macintosh 上使用。

实际上除了直接在文件名中使用子目录外，还有两种选择。

1. 最好的方法是将子目录加到 TEX 搜索路径中（见第 13.1 节）。
2. 另外一种办法是用 \graphicspath 命令来指明所用的子目录（见第 13.3 节）。不过，这比前一种方法的效率要低不少。

上述两种方法都将使 \includegraphics 自动搜索图形子目录，故可在使用

```
\includegraphics{file.eps}
```

来替代

```
\includegraphics{sub/file.eps}
```

13.1 TEX 搜索路径

因为不同的 TEX 发行版设置搜索路径的方法不完全一样，所以很难提供一个普遍适用的范例。本节所用的例子基于 Unix 下的 web2c/teTEX。尽管稍有不同，但其它的 TEX 发行版也大致采用相似的策略。

对 Unix 下的 web2c/teTEX 而言，可通过设置环境变量 TEXINPUTS 来修改 TEX 的搜索路径。如使用 csh，命令

```
setenv TEXINPUTS /dir1:/dir2:
```

会使 TEX 在搜索缺省的目录前先搜索 /dir1 和 /dir2。如果省掉最后的冒号 :，那么在搜索完 /dir1 和 /dir2 后 TEX 将不再搜索缺省的目录。如设

```
setenv TEXINPUTS :/dir1:/dir2
```

则使 TEX 在搜索缺省的目录后再搜索 /dir1 和 /dir2。而

```
setenv TEXINPUTS /dir1::/dir2
```

[30] 在现代的计算机和 TEX 环境中一般不会出现性能和效率问题，因此，插图时直接使用文件路径或者使用 \graphicspath 指明图片路径都不会有问题，同时在各主流操作系统（Windows、MacOS、Linux）上的操作也是相同的。本节的翻译仅供参考。——译注

则使 TEX 在搜索 /dir1 后接下来搜索缺省的目录，最后再搜索/dir2。

在一个目录后面加上 // 使得此目录下的所有子目录都将被搜索。例如：

```
setenv TEXINPUTS /dir1//:/dir2:
```

会使 TEX 搜索 /dir1 的所有子目录（以及子目录的子目录，等等）。使用 // 要小心，如果一目录下的文件和子目录特别多的话，它会使 TEX 的搜索速度变得很慢。

若使用 sh，可用命令

```
TEXINPUTS="/dir1:/dir2:"; export TEXINPUTS
```

来设置环境变量 TEXINPUTS。

当 LATEX 在 TEX 搜索路径中寻找文件时，不会将目录名也写到 dvi 文件中。对于旧版本的 dvips 和 xdvi 而言，由于没有搜索 TEX 的搜索路径的功能，因此，可能会找不到该文件（见第 14.4 节）。

13.2 临时改变 TEX 的搜索路径

本节讲述如何使用 Unix shell 临时修改 TEX 搜索路径，以便寻找特定项目的图像文件。用户可以为每一个 LATEX 项目创建各自不同的 shell 脚本，每一个脚本指定的路径对于该项目来说是唯一的。

例如，假设用户正在写一篇期刊文章，想要创建一个 shell 脚本 latex_paper 来代替 latex 命令。在 Unix 搜索路径上创建一个文件 latex_paper，其中包含如下内容

```
#!/bin/sh
TEXINPUTS= ~/PAPER/SUB1/:~/PAPER/SUB2/:$TEXINPUTS latex $@
```

使用如下命令将该文件转成可执行文件

```
chmod u+x latex_paper
```

设置好之后，输入

```
latex_paper file.tex
```

就会在 TEXINPUTS 的开头添加路径 /PAPER/SUB1/: /PAPER/SUB2/，之后，用 latex 运行文件 file.tex，就可以搜索 /PAPER/SUB1/ 和 /PAPER/SUB2/ 子目录下的任何图像文件。

类似地，还需要写一个 dvips_paper 脚本，以便让 dvips 在 dvi 转 ps 过程中可以搜索到图像文件。

13.3 图形文件搜索路径

缺省地，LATEX 在 TEX 搜索路径中寻找图形文件。除此之外，LATEX 还会搜索由 \graphicspath 给出的目录。例如：

```
\graphicspath{{dir1/}{dir2/}}
```

告诉 LATEX 也从目录 dir1/ 和 dir2/ 下寻找图形文件。对 Macintosh 系统来说，上面的命令改为：

```
\graphicspath{{dir1:}{dir2:}}
```

很重要的一点是，搜索由 \graphicspath 给出的目录要比由 TEXINPUTS 给出的目录慢的多。更进一步说，搜索由 \graphicspath 给出的目录要占用一定的池空间（见第 13.4节）。鉴于 \graphicspath 效率不高，所以一般不推荐使用这一命令，最好的办法就是将要使用的目录加到 TEX 搜索路径中去（见第 13.1 节）。

13.4 节约池空间

TEX 为其内部的字符串传递保留了一部分内存空间，称为**池空间**（*pool space*）。每当 TEX （试图）打开一文件，就会有一部分池空间按被永久性分配。当打开很多文件时，这种内存的占用会导致 TEX 耗光它的池空间，产生如下的错误讯息：

```
! TeX capacity exceeded, sorry [poolsize=72288]
```

由于占用的池空间与文件名的长度有关，所以其中带有子目录的文件名会加重该问题。

除了最新版的基于 web2c 的 TEX 软件和一些商业软件外，增加池空间的唯一办法就是重新编译 TEX。所幸的是，通常用下面这些节约池空间的办法就可以解决问题。

- 避免用过长的文件名。
- 不要把子目录名包括进来

  ```
  \includegraphics{images/file.eps}
  ```

  取而代之的是将子目录加到 TEX 搜索路径中或不要把图形文件放在子目录下。
- 不要使用 \graphicspath 命令。如下代码

  ```
  \graphicspath{{dir1/}{dir2/}}
  ...
  \includegraphics{file.eps}
  ```

  将使 \includegraphics 命令试图打开下列文件：

  ```
  file.eps
  dir1/file.eps
  dir2/file.eps
  ```

  每一次打开文件的尝试都会消耗池空间。应该用更改 TEX 搜索路径的办法来替代使用命令 \graphicspath。
- 给出文件全名，不要省略扩展名（例如 .eps 等）。\DeclareGraphicsExtensions（第 9.1 节）缺省定义时，命令

  ```
  \includegraphics{file}
  ```

  将使 \includegraphics 命令试图打开下列文件：

  ```
  file.eps
  file.ps
  file.eps.gz
  file.ps.gz
  file.eps.Z
  ```

若是再加上使用 \graphicspath，会导致效率特别低。

最好将 \DeclareGraphicsExtensions 中定义的扩展名列表减到最小，这样在使用省略扩展名的文件时会好些。

请注意，\includegraphics 命令只会在打开或试图打开文件时消耗池空间。由于 \includegraphics 命令打开文件的目的只是为了确定图像的 BoundingBox，因此，为了阻止池空间消耗，一种有效但却不那么方便的的方法就是，使用 \includegraphics 命令的 bb 选项指定 BoundingBox 参数值（见表 1）。

14 在 DVIPS 中使用压缩 EPS 文件和非 EPS 文件

（正如译序所述，推荐使用 XeTeX 或者 pdfTeX编译方式，因此本节一般可以略过。——译注）

正如第 1 节所述，在 dvips 模式下，LaTeX 图像插入的任务是由 dvi 程序负责。这就意味着，LaTeX 文档可以使用任何 dvi 程序支持的图像格式。尽管事实上所有的 dvi-ps 转换程序都支持 eps 图像，但几乎没有转换程序支持非 eps 图像。所以，在 dvips 模式下，要使用非 eps 图像需要将其转成 eps 格式。而这可以有两种办法。

提前转换 在 dvi-ps 转换之前，使用图像转换程序将非 eps 图像转成 eps 格式。保存的 eps 格式可以用于接下来的 dvi-ps 过程。

实时转换 在 dvi-ps 过程中，dvi-ps 转换程序调用一个图像转换程序，使得图像转换的结果通过管道传回 dvi-ps 程序，最后插入到 ps 文件中。

提前转换的缺点是需要存储图像文件的 eps 版本。尽管“实时转换”方法不需要额外存储，但需要在每一次执行 dvi-ps 程序时都进行重复的图像转换计算。所以这里需要在速度和存储之间进行权衡。不过大部分用户偏向“提前转换”带来的速度优势。

与直接整合图像转换指令不同，dvips 提供了一种调用外部转换程序的机制[31]。这可以通过在 LaTeX 中使用 \DeclareGraphicsRule 的命令选项来完成。这种方法比直接的图像转换支持更方便灵活，因为图像转换和 dvi-ps 过程是分开的，因此，用户可以选择自己想用的图像转换程序。

当在支持管道的操作系统中使用 dvips 时[32]，可以使用 \DeclareGraphicsRule （见第 9.2 节）指定在文件上执行的操作。若为解压缩命令，则可允许使用压缩的图形文件。若为图形格式转换命令，则可允许使用非 eps 图形文件。当使用不支持管道的操作系统时，这种即时转换的命令是不允许的，这时只好将所有的图形文件都存为非压缩的 eps 格式。

考虑到目前为止 dvi-ps 转换程序中只有 dvips 具有这种功能，本节所介绍的内容都需要 dvips 的支持。使用者需要在使用 graphicx 宏包时设定使用 dvips 选项。这可以通过在 \documentclass 中指定 dvips 全局选项进行全局设定：

```
\documentclass[dvips,11pt]{article}
```

或者在 \usepackage 中设定 graphicx 的使用 dvips 选项为：

```
\usepackage[dvips]{graphicx}
```

推荐使用第一种方法，因为它将 dvips 这一选项传递给所有的宏包。

[31] 这种机制需要操作系统支持管道功能。

[32] 例如，Unix 支持管道而 DOS 则不支持

14.1 压缩 EPS 文件的例子

使用压缩 eps 文件的步骤是：

1. 创建一个 eps 文件（比如说 file1.eps）。
2. 将它的 BoundingBox 存放到另外一文件中（ file1.eps.bb）。
3. 压缩 eps 文件，比如在很多平台上用命令：

```
gzip -9 file1.eps
```

得到压缩文件 file1.eps.gz。这里 -9（或者 -best）选项表示最佳压缩。

4. 在 \includegraphics前声明适当的 \DeclareGraphicsRule 命令。使得 LaTeX 知道如何处理特殊后缀的文件（见第 9.2 节）。例如：

```
\documentclass[dvips]{article}
\usepackage{graphicx}
\begin{document}
\DeclareGraphicsRule{.eps.gz}{eps}{.eps.bb}{`gunzip -c #1}
\begin{figure}
\centering
\includegraphics[width=3in]{file1.eps.gz}
\caption{压缩的 EPS 图像}
\label{fig:compressed:eps}
\end{figure}
\end{document}
```

在这个特殊的例子里，命令 \DeclareGraphicsRule 实际上是可以省略的，因为它在文件 dvips.def 中已经预定义过了。如果使用另外一个解压缩程序或文件名后缀，那么 \DeclareGraphicsRule 是不能少的。例如 BoundingBox 存放到文件 file.bb 中，则相应的 \DeclareGraphicsRule 应为：

```
\DeclareGraphicsRule{.eps.gz}{eps}{.bb}{`gunzip -c #1}
```

14.2 非 EPS 图形文件

eps 格式的图形文件可以很容易的插入到 LaTeX 文件中，而非 eps 格式的图形文件（gif、tiff、jpeg、pict 等）则不是将插图命令中的文件名替换一下就可以的。一个简单的解决方法是检查一下生成该图像的应用程序是否也可以输出 eps。如果不是的话，就必须用图像转换程序（见第 6.2 节）将其转成 PostScript。尽管使用非 eps 格式的图形文件不如 eps 图形文件简单方便，但由于它们可能比 eps 文件要小，而一些绘图软件也不能生成 EPS 文件，所以有时还是希望在 dvi 文件转换为 ps 文件时再对其进行图像格式转换。如果使用 dvips，这种即时转换的命令可用 \DeclareGraphicsRule来给出。例如用这种方法将 file2.gif 插入到 LaTeX 文档中需要以下几步：

1. 找到一个支持命令行方式的 gif 到 eps 的转换工具（假设为 gif2eps）。
2. 创建一个指定 file2.gif 自然大小的 BoundingBox 文件。为此，
   (a) 如果 PostScript 文件包含 BoundingBox 行，将此行保存到文件 file2.gif.bb

(b) 如果 PostScript 文件不包含 BoundingBox 行，用 `ebb file2.gif` 直接得到 BoundingBox 文件[33]。

(c) 将 `file2.gif` 转成 PostScript。如果 PostScript 文件包含 BoundingBox 行，则将此行保存到文件 `file2.gif.bb`；否则，可按照第 3.3 节的方法来计算 BoundingBox，并将所得到的结果放在 `file2.gif.bb` 中的 `%%BoundingBox` 后。然后将 PostScript 文件删除。

3. LaTeX 文件中，在 `\includegraphics` 命令前，加入图形规则：

```
\DeclareGraphicsRule{.gif}{eps}{.gif.bb}{`gif2eps #1}
```

当遇到 `\includegraphics{file.gif}` 时，LaTeX 会从 `file.gif.bb` 中读取 BoundingBox，并告诉 dvips 使用 gif2eps 来将 `file.gif` 转为 `eps` 文件。

14.3 GIF 的例子

由于插入非 `eps` 格式的图形所需的命令依赖于操作系统和图形格式转换程序，在此提供两个 Unix 系统下常用的转换程序的例子。

```
\DeclareGraphicsRule{.gif}{eps}{.gif.bb}{`convert #1 'eps:-' }
\begin{figure}
\centering
\includegraphics[width=3in]{file2.gif}
\caption{GIF Graphic}
\end{figure}
```

这里使用 convert 命令（包含在 ImageMagick 中）将 `gif` 转为 `eps`。而命令：

```
convert file2.gif 'eps:-'
```

将 `file2.gif` 转为 `eps` 格式（“`eps:`”选项）并输出到标准输出（“`-`”指示符）。

另一方法是使用 ppm 组件的 giftoppm、ppmtopgm 和 pgmtops 程序将 `gif` 通过 ppm 和灰度 `pgm` 格式最终转为 `eps`。在 Unix 中，使用如下 `\DeclareGraphicsRule` 命令可以将这些程序用管道连接起来：

```
\DeclareGraphicsRule{.gif}{eps}{.gif.bb}{`giftoppm #1 | ppmtopgm | pgmtops}
```

14.4 TeX 搜索路径和 dvips

当 LaTeX 遇到`\includegraphics` 命令时，会首先在当前目录下搜寻图形文件。如果找不到所需文件，则会在 TeX 搜索路径中寻找。当 `dvi` 文件转为 PostScript 文件时，dvips 也以相同顺序搜索图形文件。这不会有什么问题。然而，如果用 `\DeclareGraphicsRule` 定义了一个实时转换的命令，那么此命令将会阻止 dvips 在 TeX 搜索路径中寻找图形文件。

例如：

```
\DeclareGraphicsRule{.eps.gz}{eps}{.eps.bb}{`gunzip -c #1}
```

指定对后缀为 `.eps.gz` 的文件使用命令 `gunzip -c`。假设用下面的命令来插入图形文件，

[33] ebb 是 dvipdfm 中的一个应用程序，用来计算非 eps 图形文件的 BoundingBox。——旧译本

```
\includegraphics{file.eps.gz}
```

如果 file.eps.gz 和 file.eps.bb 在当前目录下的话，那么不需要路径搜索，一切没有问题。LaTeX 会使用 file.eps.bb 而 dvips 则执行 gunzip -c file.eps.gz 来解压缩文件。

但是，如果 file.eps.gz 和 file.eps.bb 不在当前目录就会有问题。假设在目录 /a/b/c/ 下（并且该目录已加到 TeX 搜索路径中）。LaTeX 通过搜索路径仍然能够找到 /a/b/c/file.eps.bb，但 dvips 在执行 gunzip -c file.eps.gz 就会出问题。因为 gunzip 找不到文件 file.eps.gz。

如果 TeX 发行版使用了 kpathsea 库（比如 teTeX 发行版），这个问题可用定义下面的图形规则来解决。

```
\DeclareGraphicsRule{.eps.gz}{eps}{.eps.bb}%
{`gunzip -c `kpsewhich -n latex tex #1`}
```

这里使用 kpsewhich 为 gunzip 寻找文件。

```
`kpsewhich -n latex tex #1
```

会使得 dvips 在 TeX 搜索路径中寻找压缩图形文件，然后把文件的全名（包括目录名）附加到 gunzip -c 命令后，这样即使压缩图形文件不在当前目录下，gunzip 也可对其进行操作。

虽然上面给出的新的图形规则可以放在每个 LaTeX 文件的开头，但是更方便的用法是把如下代码放到 graphics.cfg 文件中：

```
\AtEndOfPackage{%
\DeclareGraphicsRule{.eps.gz}{eps}{.eps.bb}%
{`gunzip -c `kpsewhich -n latex tex #1`}}
```

并且保留 \ExecuteOptions{dvips} 这一行。

15 在图像上添加 LaTeX 标记

15.1 PSgrag 宏包

目前很多绘图和分析软件都可以输出 eps 格式的图形，但是它们大都不能像 LaTeX 一样支持符号和公式。PSfrag 宏包允许用 LaTeX 的文本和公式来替代 eps 图形文件中的字符。

PSfrag 3.0 是 1996 年发布的完全重写版本。以前的版本则需要借助预处理程序（ps2frag 或 ps2psfrag）来识别和标记 eps 图形文件中的文本。而 PSfrag 3.0 不需要借助预处理程序，也不需要像 perl 或 ghostscript 等外部程序。PSfrag 3.0 只需要较近版本的 LaTeX 和图形宏包套件。参考文献 [5] 给出了 PSfrag 3.0 的详细说明。

PSfrag 的另一优势是支持压缩的 eps 图形。不过，\tex 命令（见第 15.1.3 节）不能用于在压缩的 eps 图形中嵌入 LaTeX 文本。

使用 PSfrag 的步骤是：

1. 在 LaTeX 文档的导言区中加入：\usepackage{psfrag}。
2. 在文档中使用 \psfrag 命令来指明要替换的 eps 文本以及取而代之的 LaTeX 字符串。同一环境下之后的任何 \includegraphics 命令中会执行这些替换。

表 5: PSfrag 选项

PStext	eps 图形中被替换的文本。
posn	（可选项，缺省为 [Bl]）放置点相对于 LaTeX 文本的参考位置。
PSposn	（可选项，缺省为 [Bl]）放置点相对于现有 eps 文本的参考位置。
scale	（可选项，缺省为 1）LaTeX 文本的缩放因子。为得到最好的效果，建议不使用这一选项，而使用 \small 和 \large 等 LaTeX 字体命令。
rot	（可选项，缺省为零）当给出一个角度时，此角度即为新的 LaTeX 文本相对于旧的 eps 图形中文本的角度。它以度为单位并且逆时针方向为正。此选项特别适用于应用软件只能生成包含水平方向文本的 eps 图像的情形。
text	用来插入 eps 图像的 LaTeX 文本。如同通常的 LaTeX 文本，数学公式必须放在美元符号对中。如 $\frac{1}{2}$ 或 x^2。

3. 像通常一样使用 \includegraphics 即可。

\psfrag 命令的用法如下：

```
\psfrag{PStext}[posn][PSposn][scale][rot]{text}
```

其中选项说明见表 5。

选项 posn 和 PSposn 可以是第 32 页图 3 所示的 12 个点中的一个（例如 [tl]、[br]、[cc]）。如果没有给出，默认点是 [Bl]。缺失字母默认为 c，例如 [] 和 [c] 都等价于 [cc]，[l] 等价于 [lc]。可参考 [5] 中各种位置组合的例子。

需要注意的是 \psfrag 只匹配整个字符串，例如下面的命令

```
\psfrag{pi}{$\pi$}
```

用 π 替换 pi，但不会影响其它像 pi/2 或 2pi 这样的 eps 字符串。对于必须各自使用 \psfrag 命令。

间距问题

如果所替换的 eps 字符串不是完整的置于一个 PostScript 命令中，那么 \PSfrag 将不起作用。在一些应用软件生成的 eps 图形中，为达到特殊的字符间距，将一字符串分隔为几个子串或单个的字符。例如，Corel Draw 可以用如下的 eps 代码来放置字符串“Hello World”：

```
0 0 (Hello W) @t
1080 0 (orld) @t
```

由于 PSfrag 将其视为是两个不相干的字符串“Hello W”和“orld”，所以任何对“Hello World”的替换都不起作用。如果不能在应用软件中手动取消这种对字符间距的处理，一般可以使用 Courier 或其它等宽字体来阻止这种情况[34]。如果确实无法避免这种情况，那么只能对单个字符进行替换。

[34] 为了避免字符间距，可能需要在创建文本之前设置 Courier 字体。先穿件文本在将其转成 Courier 字体可能仍然会有字符间距。

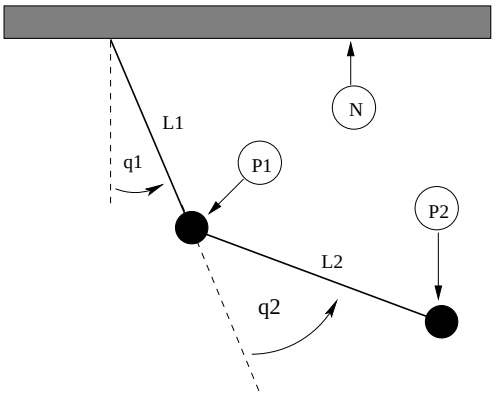


图 9: PSfrag 替换前

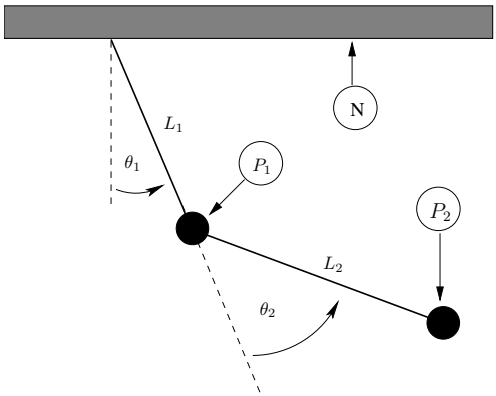


图 10: PSfrag 替换后

15.1.1 PSfrag 使用例 #1

命令

```
\includegraphics{pend.eps}
```

只是插入 eps 图形而没有任何 psfrag 替换，见图 9。而下面的命令

```
\psfrag{q1}{$\theta_1$}
\psfrag{q2}{$\theta_2$}
\psfrag{L1}{$L_1$}
\psfrag{L2}{$L_2$}
\psfrag{P1}[][]{$P_1$}
\psfrag{P2}[][]{\large $P_2$}
\includegraphics{pend.eps}
```

在插入 eps 图形的同时使用 psfrag 对 eps 图形中的字符串进行替换，见图 10。前四个 \psfrag 命令中，新的 LaTeX 字符串的左基线点对应于旧的 eps 字符串的左基线点，后面两个 \psfrag 命令中使用 [][] 选项使得新的 LaTeX 字符串的中心对应于旧的 eps 字符串的中心。注意并不是所有的 eps 字符串都被替换，如在图 10 中 N 就没有被替换。

15.1.2 PSfrag 使用例 #2

这个例子演示了 \shortstack, \colorbox 和 \fcolorbox 等命令如何与 \psfrag 一起使用。

\shortstack 这一命令允许将文本竖直放置，每行用 \\ 分开。可用来使用多行文本替换图中的单行文本。

\colorbox color 宏包所提供的一个命令，它在所作用的对象背后放置长方形的彩色区域作为背景。此背景超出对象的部分的大小由长度 \fboxsep 控制。例如：

```
\colorbox{yellow}{文本}
```

在 文本 后放置了一长方形的黄色背景。有关 \colorbox 的详细说明可参考文献 [1]。

在使用 PSfrag 时，\colorbox 常常用来放置那些由于线条或阴影而被遮挡的文本。通过将这些文本的背景色设为白色，防止它们被图形所遮挡。

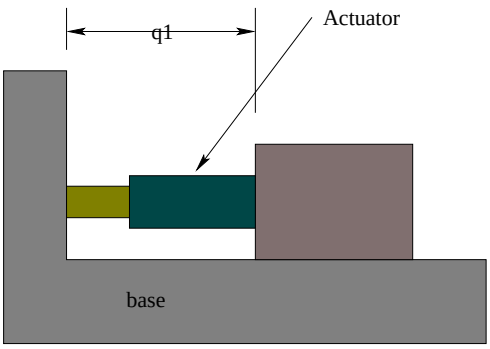


图 11: PSfrag 替换前

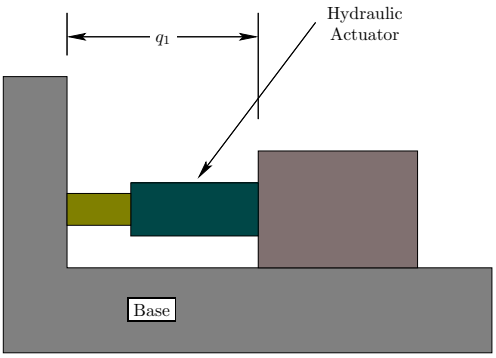


图 12: PSfrag 替换后

\fcolorbox 命令（也由 color 宏包所提供）与 \colorbox 类似，只是为背景加上了一个边框。例如命令

```
\fcolorbox{black}{yellow}{文本}
```

在 文本 后放置了一长方形带有黑色边框的黄色背景。

这里边框的宽度由 \fboxrule 控制，边框和对象之间的间隔大小则由 \fboxsep 控制。

图 11 是没有使用 PSfrag 的原始图形，而图 12 则是使用如下命令的结果。

```
\psfrag{q1}[][]{\colorbox{white}{$q_1$}}
\psfrag{base}{\fcolorbox{black}{white}{Base}}
\psfrag{Actuator}[l][l]{\shortstack{Hydraulic\\ Actuator}}
\includegraphics{mass.eps}
```

下面的例子使用了中文，结果如图 13。注意，由于最新的 ctex 宏集不支持自动配置 dvips，需要手动使用 CJK 或配置字体文件。

```
%\usepackage{CJKutf8}
\begin{CJK}{UTF8}{gkai}
\psfrag{q1}[][]{\colorbox{white}{$q_1$}}
\psfrag{base}{\fcolorbox{black}{white}{基础部分}}
\psfrag{Actuator}[l][l]{\shortstack{水力\\ 驱动器}}
\includegraphics{mass.eps}
\end{CJK}
```

15.1.3 EPS 图形中的 LaTeX 文本

在使用 PSfrag 宏包时，\psfrag 命令是最常使用也是推荐使用的。它的具体用法已在前面几小节中介绍过了。此外，PSfrag 宏包还提供 \tex 命令直接将 LaTeX 文本嵌入到 eps 中。不过，它的效率要比 \psfrag 低。有关 \tex 的详细信息可参见 [5]。

15.1.4 图形和文本的缩放

如果缩放一幅使用了 PSfrag 的图形，那么其中 PSfrag 所替换的文本也相应的被缩放。因此，使用 graphicx 包时的一些细节有可能影响到这些文本的大小。

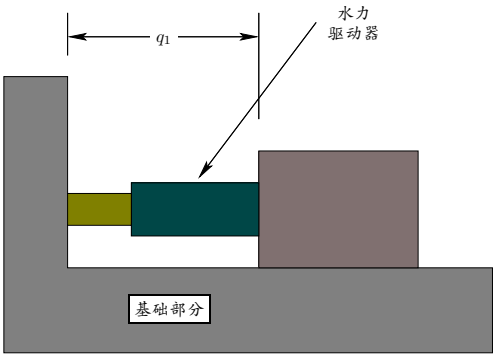


图 13: PSfrag 中使用中文的例子

- 当使用 width、height 或 totalheight 指定图形的大小时，如

  ```
  \includegraphics[width=3in]{file.eps}
  ```

  PSfrag 会在缩放之后再插入文字。相反地，

  ```
  \resizebox{3in}{!}{\includegraphics{file.eps}}
  ```

  则先将图像以它的自然大小插入，然后进行 PSfrag 替换，最后再将图形和所替换的文本一起缩放。

- 相似地，当缩放选项在旋转选项之前时，

  ```
  \includegraphics[width=3in,angle=30]{file.eps}
  ```

  缩放选项会得到预期的效果。然而，当它在旋转选项之后时，

  ```
  \includegraphics[angle=30,width=3in]{file.eps}
  ```

  图像会先以其自然大小插入，然后被旋转，接着进行缩放。因为 PSfrag 的替换是在图形被插入时发生的，所以第二个命令中的替换文本会被缩放，而第一个命令中的替换文本不会被缩放。如果图形的自然大小和被缩放后的大小差别很大的话，这两个命令得到的结果会大不相同。

参见 [5] 以获取关于 PSfrag 文本缩放的详细说明。

15.1.5 PSfrag 和 pdfTeX

psfrag 不能在 pdfTeX 模式下使用。如果需要进行 psfrag 替换，必须间接地使用 psfrag 进行图像，可以使用 pstool 宏包[35]。

pstool 宏包会自动导入 psfrag 宏包。它提供的 \psfragfig 命令可以对 eps 图像进行实时 psfrag 替换，并使用 latex-dvips 生成相应的 pdf 格式图片，然后插入到文档中。使用方法为

\psfragfig[⟨*option*⟩]{⟨*filename*⟩}{⟨*input definitions*⟩}

其中，⟨*option*⟩ 可以是 \includegraphics 命令的可选项，例如 angle、width 等。⟨*filename*⟩ 是图片的文件名，注意不要包含 .eps 后缀。⟨*input definitions*⟩ 则包含 \psfrag 定义。更详细的说明请参考宏包文档 [15]。

[35] 本节由译者添加。

15.1.6 PSfrag 和其它编译方式

dvipdfmx、XeTeX、LuaTeX 等其它编译引擎不支持 psfrag 和 pstool，因此此时只能手动转换每一幅要使用 psfrag 的图像。具体步骤为

1. 对于每一个要使用 psfrag 的图像，分别创建包含 psfrag 命令和 \includegraphics 命令的 LaTeX 文件。其中注意使用 \pagestyle{empty} 以禁止页码的出现。假设这些 LaTeX 文件为

   ```
   GraphicFrag00.tex
   GraphicFrag01.tex
   ...
   ```

2. 在操作系统的命令行中执行以下步骤[36]：

   ```
   latex GraphicFrag00.tex
   dvips -E GraphicFrag00
   ps2eps -B GraphicFrag00.ps
   epstopdf GraphicFrag00.eps
   ```

   第一行命令创建文件 GraphicFrag00.dvi。第二行命令创建 GraphicFrag00.ps。第三行命令为该 ps 文件计算 BoundingBox，并将 BoundingBox 和 GraphicFrag00.ps 的内容导入 GraphicFrag00.eps。最后一行命令将 GraphicFrag00.eps 转成 pdf 格式。

3. 重复第二步的内容依次处理 GraphicFrag01.tex 等等。

4. 在原来的 LaTeX 文件中使用 \includegraphics 命令导入生成的 pdf 图像

   ```
   GraphicFrag00.pdf
   GraphicFrag01.pdf
   ...
   ```

5. 使用 latex+dvipdfmx、xelatex、lualatex 等编译程序处理 LaTeX 文件。

15.2 Overpic 宏包

尽管 PSfrag 的功能十分强大，使用起来也很方便，但只适用于 eps 图像并使用 dvips 引擎。但对于非 eps 图形或标记并非标准字符串的 eps 图来说，它就不能被使用。此外，一些 TeX 程序如 dvipdfm(x)、pdfLaTeX 等不能直接使用 PostScript 替换，也限制了 PSfrag 的使用。本节所介绍的 overpic 宏包允许直接将 LaTeX 对象放置到一幅图形上，而不是通过对图形上已有的标记进行替换来实现。这样，虽然在定位时要麻烦一些，却可以在一些不能使用 PSfrag 的情况下得到同样的效果。

overpic 宏包定义的 overpic 环境能有效地将 picture 环境和 \includegraphics 命令结合起来。这样使得 picture 环境和插入图像的大小相同，进而可以很容易地把 LaTeX 的命令放到图形上的任何指定位置。同时，还可以在图形上加上标尺以方便定位。

overpic 环境的用法为

[36] 这里使用 TeXLive 自带的 ps2eps 程序代替原文中的第三方程序 epstool。——译注

```
\begin{overpic}[⟨选项⟩]{⟨图形⟩}
    ⟨LaTeX 对象⟩
\end{overpic}
```

这里的 ⟨选项⟩ 可以是 \includegraphics 的可选项，此外还包括

- grid 是否加标尺。
- tics= 标尺的刻度值。
- unit= 标尺的单位。

在调入 overpic 宏包时，若使用参数 abs，即

```
\usepackage[abs]{overpic}
```

则在 overpic 环境中使用绝对位置，即放置 LaTeX 对象的位置以实际度量来定位。此时 overpic 选项的默认值为 tics=10，unit=\unitlength[37]

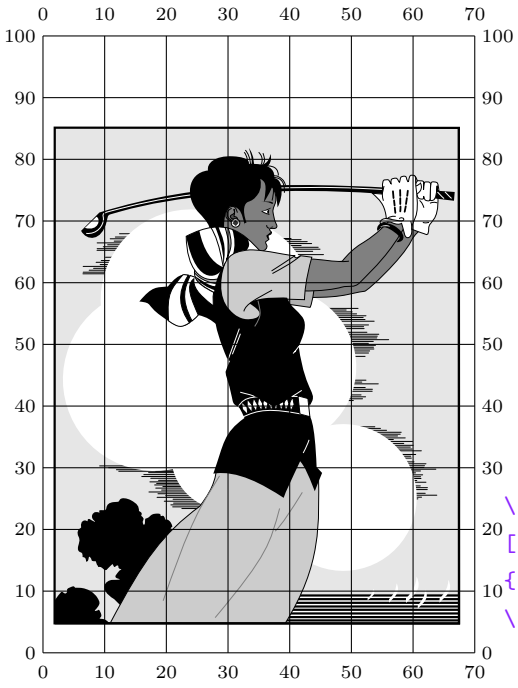
若使用

```
\usepackage[percent]{overpic}%默认值，等价于\usepackage{overpic}
```

或者

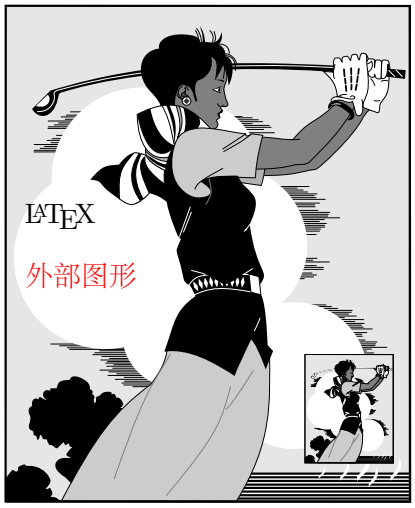
```
\usepackage[permil]{overpic}
```

则在 overpic 环境中使用相对位置，即放置 LaTeX 对象的位置以其相对于图形大小的百分比来定位。对于 percent 宏包选项（默认值），相关设置为 tics=10，unit 为图像较长一边长度的 1/100；对于 permil 宏包选项，相关设置为 tics=100，unit 为图像较长一边长度的 1/1000。下面是几个例子（使用默认的 percent 相对位置）：




```
\begin{overpic}%
[width=\linewidth,grid,tics=10]%
{golfer}
\end{overpic}
```

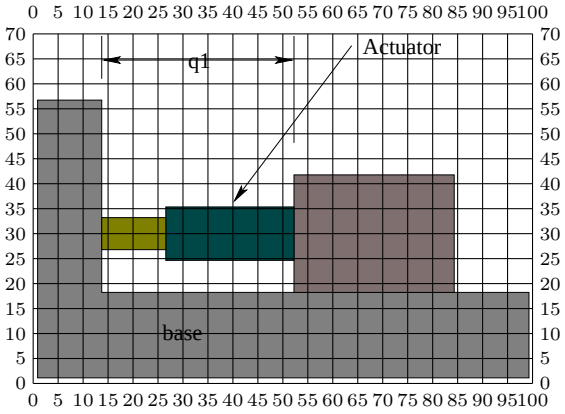
[37] \unitlength 为 picture 环境的长度单位，在 LaTeX 中默认设置为 1 pt。




```
\begin{overpic}[width=\linewidth]{golfer}
\put(5,50){\LaTeX}
\put(5,40){\color{red}外部图形}
\put(50,10){%
\includegraphics[scale=.07]%
{golfer}}
\end{overpic}
```

对于第 15.1.2 节中的例子，现在使用 `overpic` 宏包来得到同样的结果。首先可使用 `grid` 和 `tics` 选项来确定放置 LaTeX 对象的位置（这样做只是为了能够得到更精确的放置位置，在定位后就可将 `grid` 和 `tics` 选项去掉）。

```
\begin{overpic}[scale=1.2,grid,tics=5]{mass}
  \end{overpic}
```



根据上图，将所需的 LaTeX 对象放到图形上的合适位置：

```
\begin{overpic}[scale=1.2]{mass}
  \put(25,8){\fcolorbox{black}{white}{基础部分}}
  \put(31,64){\colorbox{white}{$q_1$}}
  \put(65,65){\colorbox{white}{\shortstack{水力 \\ 驱动器}}}
\end{overpic}
```

结果如图 14 所示。

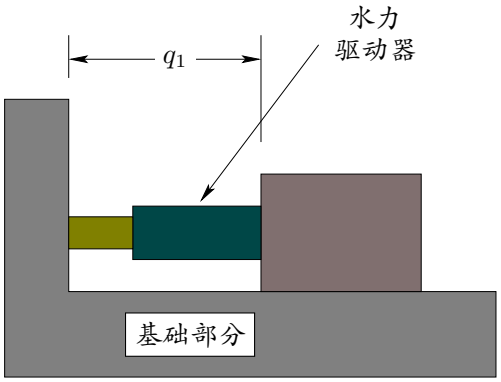


图 14: Overpic 例子

16 多次使用同一图形的几种技巧

在文档的页眉或页脚使用标志或其它图形时就会遇到多次使用同一图形的情况。特别地，当使用 dvips 处理文档时，如果多次使用同一 eps 图像，那么它的 eps 代码就会多次出现在得到的 ps 文件中。本节将介绍一些相关的技巧。[38]

一般来说多次使用同一图形有以下一些方法：

1. 每次使用图形时均用 `\includegraphics{file}`。但是每次使用 `\includegraphics` 时 LaTeX 都得搜索和打开一次图形文件。

2. 将 eps 图形文件存放到一个 LaTeX 盒子中，每当用到图形时就调用这个 LaTeX 盒子来插入图形。这样 LaTeX 只需搜索和打开一次图形文件即可。在 LaTeX 文件加入命令：

```
\newsavebox{\mygraphic}
\sbox{\mygraphic}{\includegraphics{file}}
```

   每次使用图形时，用命令 `\usebox{\mygraphic}` 即可。另外，将 `\usebox` 命令放在 `\scalebox` 或者 `\resizebox` 中即可实现图形的缩放和旋转。

16.1 在 DVIPS 模式下使用 Postscript 命令插入 EPS 文件

对于 dvips 模式下多次使用同一 eps 文件的文档，即使使用 LaTeX 盒子避免多次读取同一图像文件，但是在最后生成的 ps 文件中，eps 图形代码仍会多次出现，所以生成的 ps 文件仍然很大。对此有以下几种方法 [39]

1. 当 eps 文件只包含矢量图形时，可将此绘图的代码定义为一个 PostScript 命令，当用到图形时就调用这一个 PostScript 命令[40]。因为在最后生成的 PostScript 文件中只包含了一次 eps 的图形代码，所以 PostScript 文件会很小。不过需要注意的是，

[38] 本节结构编排与原文略有出入。——译注

[39] 用户还可以考虑使用 graphicx-psmin 宏包。该宏包可以使得多次使用的 PostScript 图像也仅需导入一次。限于篇幅本文档不做介绍。

[40] 尽管可以构建 PostScript 命令画矢量图，但这对于位图是不可能的。位图在转成 eps 格式时，一般都使用 image （或者 colorimage）PostScript 运算符直接读取当前文件作为数据。因此，该过程不可以使用 PostScript 直接画矢量图形。当然，可以修改 eps 文件，使得图像数据作为 PostScript 字符串进行传递而不是直接读取二进制文件。但是很难进行自动化操作，需要大量的手动编辑 PostScript。由于大部分人并不是 PostScript 专家，所以手写 PostScript 并不在考虑范围内。如果图像可以用 PostScript 矢量原语描述，可以考虑使用 kvec 程序（见第 6.2 节）将图像转成矢量 PostScript 格式。

在打印时由于绘图命令一直存放在打印机的内存中，很容易导致打印机的内存耗尽而无法打印。另外，使用这种方法时，LaTeX 仍然得每次对图形文件进行搜索和打开操作。

2. 像之前方法一样定义 PostScript 绘图命令，但把它存放到一个 LaTeX 盒子中。这样不仅最后生成的 PostScript 文件很小，而且 LaTeX 也只需搜索和打开一次图形文件即可。

本节介绍如何定义 PostScript 命令来完成一幅 eps 矢量图的绘图指令。这一方法不适合于那些包含位图的 eps 图形。

为了将 eps 图形转化为 PostScript 命令，必须将 eps 图形文件分为两个文件。其中一个定义了 PostScript 字典和图形命令，另一个则含有图形文件信息和使用已定义的 PostScript 命令。例如，一个用 Xfig 生成的 eps 文件有如下的形式：

```
%!PS-Adobe-2.0 EPSF-2.0
%%Title: /tmp/xfig-fig017255
%%Creator: fig2dev Version 2.1.8 Patchlevel 0
%%CreationDate: Sun Sep 3 15:36:01 1995
%%Orientation: Portrait
%%BoundingBox: 0 0 369 255
%%Pages: 0
%%EndComments
/$F2psDict 200 dict def
$F2psDict begin
...
%%EndProlog
$F2psBegin
...
$F2psEnd
```

eps 文件一般包括三部分：

1. 以 % 开始的文件头命令。

2. Prolog 部分，开始于

   ```
   /$F2psDict 200 dict def
   ```

   结束于

   ```
   %%EndProlog
   ```

   Prolog 部分在 PostScript 字典中定义了 eps 文件所使用的命令。在这个例子中，PostScript 字典名为 $F2psDict，当然不同的文件可有不同的名字。

3. 最后一部分包含用来绘图的命令。

假设上面的这个 eps 文件名为 file.eps。新建两个文件，分别命名为 file.h 和 file.ps。其中 file.h 含有如下内容：

```
/$F2psDict 200 dict def
$F2psDict begin
...
%%EndProlog
/MyFigure {
$F2psBegin
...
$F2psEnd } def
```

file.ps 含有如下内容:

```
%!PS-Adobe-2.0 EPSF-2.0
%%Title: /tmp/xfig-fig017255
%%Creator: fig2dev Version 2.1.8 Patchlevel 0
%%CreationDate: Sun Sep 3 15:36:01 1995
%%Orientation: Portrait
%%BoundingBox: 0 0 369 255
%%Pages: 0
%%EndComments
$F2psDict begin MyFigure end
$
```

file.h 中定义了 PostScript 字典和命令 /MyFigure，file.ps 则包含了文件头部信息，并且使用了 file.h 中定义的 PostScript 命令。特别指出的是，file.ps 中包含有 %!PS... 这一行和 BoundingBox 行，这是非常重要的。这时，可像下面的例子一样在 LaTeX 文件中使用这个图形了。

```
\documentclass{article}
\usepackage{graphicx}
\special{header=file.h}
\begin{document}
    ...
    \includegraphics[width=2in]{file.ps}
    ...
    \includegraphics[totalheight=1in]{file.ps}
    ...
\end{document}
```

注意这里并没有使用原始图形文件 file.eps。因为 file.h 中的 PostScript 命令只导入一次，所以最后得到的 PostScript 文件很小。然而，每次插入图形的时候，LaTeX 都要搜索和读取一次 file.ps。下面的命令将图形存放到一个 LaTeX 盒子中，使得在 LaTeX 只搜索和读取一次 file.ps 的情况下仍得到很小的 PostScript 文件。

```
\documentclass{article}
\usepackage{graphicx}
\special{header=file.h}
\newsavebox{\mygraphic}
\sbox{\mygraphic}{\includegraphics[width=2in]{file.ps}}
\begin{document}
```

```
    ...
    \usebox{\mygraphic}
    ...
    \resizebox*{1in}{!}{\usebox{\mygraphic}}
    ...
\end{document}
```

和上一个例子一样，最后会生成一幅宽度为 2 英寸的图像和一幅总高度为 1 英寸的图像。

16.2 在页眉和页脚使用图形

在页眉和页脚使用图形的一个最容易的方法是使用 fancyhdr 宏包（它是过时宏包 fancyheadings 的增强版本）。fancyhdr 的用法和宏包说明详见说明文档 [13]。

在 LaTeX 文档中，页眉由左、中、右三部分组成。\fancyhead 命令指定了页眉的形式和内容，并以 L,C,R 区分左、中、右区域。例如：

```
\pagestyle{fancy}
\fancyhead[C]{我的文档}
```

使得页眉的中间部分印出“我的文档”。

```
\pagestyle{fancy}
\fancyhead[L,R]{\textbf{Confidential}}
```

使得页眉的左右都印出“**Confidential**”。如果没有指定 L,C,R，那么在三个区域中都会印出由 \fancyhead 定义的内容。所以，可以用 \fancyhead{} 来清空页眉。相似地，\fancyfoot 则用来定义页脚的左、中、右三个区域。

页眉和页脚的图形

可以利用 fancyhdr 宏包中的命令 在页眉和页脚中插入图形。例如，在用第 16.1 节的方法将 eps 文件 file.eps 分为两个文件 file.h 和 file.ps 后，下面的命令

```
\documentclass{article}
\usepackage{fancyhdr,graphicx}
\renewcommand{\headheight}{0.6in} %% must be large enough for graphic
\renewcommand{\textheight}{7.5in}
% Define PostScript graphics command
\special{header=file.h}
% Save graphics in LaTeX box
\newsavebox{\mygraphic}
\sbox{\mygraphic}{\includegraphics[totalheight=0.5in]{file.ps}}
\pagestyle{fancy}
\fancyhead{} % clear all header fields
\fancyhead[L]{\usebox{\mygraphic}}
\fancyfoot{} % clear all footer fields
\fancyfoot[C]{\thepage}
\renewcommand{\headrulewidth}{0.5pt}
\renewcommand{\footrulewidth}{0pt}
\begin{document}
    ...
\end{document}
```

在每一页面风格为 fancy 的页面上，图形会放置在页面的左上角，并且下面有一条宽为 0.5 pt 的横线。此外，页角的中央放置页码，但它的上方没有横线。注意这些设置不会影响 plain 风格的页面。

奇/偶页的页眉

当使用 [twoside] 文档类选项时，经常希望在奇数页和偶数页设置不同页眉和页脚，这时可使用 O,E 选项来区分奇数页和偶数页。如果没有给出 O,E 选项，那么命令会应用到所有的页面中，无论是奇数页还是偶数页。类似地，\fancyfoot 命令的 O,E 选项用于设置奇数页和偶数页的页脚。例如：

```
\pagestyle{fancy}
\fancyhead[LE]{我的文章}
\fancyhead[RO]{我的名字}
\fancyfoot[C]{\thepage}
```

在偶数页的左上角放置“我的文章”，在奇数页的右上角放置“我的名字”，页脚的中央则放置页码。

在之前例子中将

```
\fancyhead[L]{\usebox{\mygraphic}}
```

替换为

```
\fancyhead[LE,RO]{\usebox{\mygraphic}}
```

那么图像将出现在所有 fancy 风格页面的页眉外侧，即奇数页的右侧和偶数页的左侧。

改变 plain 页面样式

\fancyhead 命令只对 fancy 样式的页面起作用。即使使用命令 \pagestyle{fancy} 将文档的页面样式设置为 fancy，但封面、目录和每章的第一页等页面仍为缺省的 plain 样式。

改变 plain 样式的缺省设置可用 \fancypagestyle 命令来实现。例如将下面的命令加到上面的例子中可使得 plain 样式页面的页眉上也印出图形。

```
\fancypagestyle{plain}{%
    \fancyhead{} % clear all header fields
    \fancyhead[L]{\usebox{\mygraphic}}
    \fancyfoot{} % clear all footer fields
    \fancyfoot[C]{\thepage}
    \renewcommand{\headrulewidth}{0.5pt}
    \renewcommand{\footrulewidth}{0pt}}
```

当使用 [twoside] 文档类选项时，将上面的

```
\fancyhead[L]{\usebox{\mygraphic}}
```

替换为

```
\fancyhead[LE,RO]{\usebox{\mygraphic}}
```

则每一页的页眉的外侧都印出图形。

16.3 在背景中使用图形水印

除了在页眉页脚处插入图像外，fancyhdr 宏包还可以将图像放在文本下层，这样可以用于创建 logo 或者水印。

下面的例子将 file.eps 图像放在每一页上（包括 fancy 和 plain 页面样式）。

```
\documentclass{article}
\usepackage{graphicx,fancyhdr}
%%% store graphics in a box
\newsavebox{\mygraphic}
\sbox{\mygraphic}{\includegraphics[keepaspectratio,
    height=0.8\textheight,
    width=0.8\linewidth]{file.eps}}
\pagestyle{fancy}
\fancyhead{}
\fancyhead[C]{\setlength{\unitlength}{1in}
    \begin{picture}(0,0)
    \put(-2.2,-6){\usebox{\mygraphic}}
    \end{picture}}
\fancypagestyle{plain}{%
    \fancyhead{}%
    \fancyhead[C]{\setlength{\unitlength}{1in}
        \begin{picture}(0,0)
        \put(-2.2,-6){\usebox{\mygraphic}}
        \end{picture}}}
\begin{document}
    ...
\end{document}
```

在这个例子中，图像的左下角位于页眉中心下方 6 英寸，偏左 2.2 英寸的地方。可以通过改变这两个数字调整图像的位置。

因为页眉在正文文本之前排出，所以正文文本会覆盖图形，从而得到水印的效果。反之，因为页脚在正文文本之后排出，所以若在页脚中使用图形会覆盖正文文本。如果 `file.eps` 是矢量图而不是位图，可用第 16.1 节的方法来使得最后生成的 PostScript 文件较为小些。

16.3.1 使用 eso-pic 宏包

另一种得到图形水印效果的方法是 `eso-pic` 宏包。它定义一个长度为零的 `picture` 环境，基点位于页面的左下角。[41]

它提供命令 `\AddToShipoutPicture` 把任何 LaTeX 图形环境先于正文文本排出，从而得到水印的效果。以下是一个使用 `eso-pic` 的例子：

```
\begin{document}
\usepackage{graphicx}
\usepackage{eso-pic}

%%% store graphics in a box
\newsavebox{\mygraphic}
  \sbox{\mygraphic}{%
    \includegraphics[keepaspectratio, height=\textheight,%
                     width=\textwidth]{file.eps}}
\AddToShipoutPicture{
    \put(0,0){\usebox{\mygraphic}}}
```

[41] 原文只提及该宏包，以下部分由译者扩充。——译注

```
\begin{document}
  ...
\end{document}
```

上面的例子中，将 eps 图形放大到与正文一样大小，然后存放到一个 LaTeX 盒子中。在每一页中将此盒子放置在正文的底层。更多命令和选项请参考宏包说明文档 [12]。此外，实现水印效果的宏包还有 background [8]、xwatermark [11] 等，详细说明请参考各自宏包文档。

第四部分　LaTeX 浮动图形环境

17　浮动图形环境

在使用字处理软件排版时，用户放在哪里图形就会出现在哪里[42]。但是，因为这些图形不能被分割开来，所以经常会导致糟糕的分页，将大片的空白留在页面下方。为得到专家级的排版效果，作者不得不手工调整图形的位置。这种工作是非常乏味的，尤其是几乎每次修改文档都得这样做一次。

为此，LaTeX 提供了一个浮动图形机制来自动将图形放置到看起来合适的位置，既能得到专家级的排版效果，又不必手工做调整图形位置的乏味的工作。不过，它也会给那些习惯于手工调整图形的新手带来不适。想要有效的利用浮动图形机制需要注意以下几点：

不要使用依赖于图形放置位置的文本　使用如“这幅图...”或“下面的图形...”等短语要求所指的图形需在固定位置。而像“图 5...”这样的短语则允许图形出现在任意位置。

放松　一些用户在发现图形没有十分准确的出现在他们所想要的位置时，往往非常着急。这没有必要，图形的放置是 LaTeX 的工作，最好放松一些。

经验总结

在接下来的几页中我们将介绍 LaTeX 是以怎样的专业级的排版规则来决定浮动图形的位置的。为方便起见，下面列出关于浮动图形放置的一些最常见问题的解决方法。

1. 不要束缚 LaTeX 的手脚。给出的浮动选项越多，LaTeX 做的就越好。具体来说，使用选项 `[htbp]` 和 `[tbp]` 就很好。见第 17.2 节。

2. 很多人发现缺省的浮动参数过于严格了。下面的命令

```
\setcounter{topnumber}{4}
\setcounter{bottomnumber}{4}
\setcounter{totalnumber}{10}
\renewcommand{\textfraction}{0.15}
\renewcommand{\topfraction}{0.85}
\renewcommand{\bottomfraction}{0.70}
\renewcommand{\floatpagefraction}{0.66}
```

   将浮动参数重新设置为更宽松的值。详见第 18 节。

3. LaTeX 允许图形浮动到当前页的顶部，这样会使图形在引述它的文本前出现。不喜欢这样做的用户可以使用 `flafter` 宏包。无需使用特殊的命令，只要简单地调入该宏包 `\usepackage{flafter}` 即可。

4. 要确保图形的浮动不超过某一特定点，可调入 `placeins` 宏包并使用 `\FloatBarrier` 命令。见第 17.3 节。

   警告：过多使用 `\FloatBarrier` 命令会导致浮动位置难以控制或浮动参数不正确。这两种情况都是应当尽量避免的。

[42] 尽管许多字处理软件确实可以允许图形在文本周围移动（或者文本在图形周围移动），但是出于糟糕的软件设计以及/或者文档作者的忽略，绝大部分用户并不使用该功能。

17.1 创建浮动图形

可以通过把命令置于 figure 环境中来生成浮动图形。在图形环境中的所有内容都会被保持在一起，浮动到合适的位置以保证得到最好的分页结果。另外，可以通过使用 \caption 命令来为浮动图形自动编号并加上标题。例如，下面的命令将 graph.eps 中的图像放到一个浮动图形中。

```
\begin{figure}
\centering
\includegraphics[totalheight=2in]{graph.eps}
\caption{一幅插入的EPS图像} \label{fig:graph}
\end{figure}

第~\pageref{fig:graph} 页的图~\ref{fig:graph}...
```

对于图形环境，应当注意：

- 可选的 \label 命令和 \ref、\pageref 命令配合使用，可对图形标题进行交叉引用。更多信息参见第 17.1.1 节。\label 命令必须紧接着 \caption 命令给出。如果放在之前的话，那么 \ref 会引用上一个可引用对象（通常是所在章节或者是之前的图形）。
- 如果 figure 图形环境中没有使用 \caption 命令，那么就是一个没有编号的浮动图形。
- 如果一图形环境中使用了多个 \caption 命令，那么它将生成多个一起浮动的图形。这在排版并列放置的图形（见第 28 节）或者复杂排列的图形（见第 31 节）时是非常有用的。
- 可用命令 \listoffigures 来得到一个图形目录。
- 缺省地，\caption 的文本就是标题内容，也会在图形目录中列出。\caption 命令的可选项可用来将与标题文本不同的内容加到图形目录中。如：

  ```
  \caption[List Text]{Caption Text}
  ```

  会在标题中使用 Caption Text 而在图形目录中使用 List Text。这在使用了特别长的标题时会很有用。
- figure 图形环境只能用于段落外模式，不能在段落中使用。因此不能放在任何的盒子中（例如 \parbox 或 minipage）。
- 若 figure 图形环境被置于正文段落中，例如

  ```
  ....text text text text text text
  \begin{figure}
  ....
  \end{figure}
  text text text text text text...
  ```

  那么它在该正文段落结束之前不会被处理。

17.1.1 定义引用命令

相比于直接输入

```
第~\pageref{fig:graph} 页的图~\ref{fig:graph}
```

更为方便的做法是在文档的导言区定义如下命令

```
\newcommand\Figpage[1]{第~\pageref*{#1} 页的图~\ref*{#1}}
```

然后引用代码就可以用 `\Figpage{fig:graph}` 来简化。

条件引用 以上定义的 `\Figpage` 命令总是会打印出图形编号和页码。当图形和引用出现在同一页时，通常需要省略页码。使用如下代码即可

```
\newcommand\FigDiff[1]{第~\pageref{fig:graph} 页的图~\ref{fig:graph}}
\newcommand\FigSame[1]{图~\ref{fig:graph}}
\newcommand\Figref[1]{\ifthenelse{\value{page}=\pageref{#1}}
    {\FigSame{#1}}{\FigDiff{#1}}}
```

如果引用和图形在同一页，那么 `\Figref` 命令会调用 `\FigSame` 命令进而显示“图 17”字样。如果不在同一页，那么 `\Figref` 命令会调用 `\FigDiff` 命令进而显示“第 25 页的图 17”字样。

宏包 varioref [9] 还为引用图表、章节等提供了类似的命令。

17.1.2 hyperref 宏包

hyperref 宏包允许用户为 LaTeX 文档创建超链接，主要结合 pdflatex 一起使用。

hyperref 的特性之一就是重定义了 `\ref` 和 `\cite` 命令，这样可以作为指向引用的超链接排版显示。同时，`\ref*` 和 `\cite*` 定义为不含超链接的引用。

由于 hyperref 重定义了很多 LaTeX 命令，通常用户需要注意 `\usepackage` 的顺序以使得最后导入 hyperref 宏包。更多信息请参考 [14]。

超文本引用 当使用 hyperref 宏包时，`\ref` 命令会打印出的图形编号会带有指向该图形的超链接。由于图形编号数字相对较小，会给点击超链接带来困难，因此，如下定义的命令会让点击超链接更加容易

```
\newcommand\Figlink[1]{%
    \hyperref[#1]{第~\pageref{fig:graph} 页的图~\ref{fig:graph}}
```

使用 `\Figlink` 命令会将“图 17”这样的引用整个地放在超链接里面。

17.2 图形的放置

利用 figure 环境的可选参数项可以指定图形有可能放置的位置。这一可选参数项可以是下列字母的任意组合。

h 当前位置。 将图形放置在正文文本中给出该图形环境的地方。如果本页所剩的页面不够，这一参数将不起作用。

t 顶部。 将图形放置在页面的顶部。

b *底部*。 将图形放置在页面的底部[43]。

p *浮动页*。 将图形放置在允许有浮动体的页面上。

几个注记：

- 如果在图形环境中没有给出上述任一参数，则缺省为 `[tbp]`。该默认选项可以通过重定义内部命令 `\fps@figure` 定制。例如，

```
\makeatletter
\def\fps@figure{htbp}
\makeatother
```

会将缺省值设为 `[htbp]`。

- 位置参数的顺序不会影响到最后的结果。因为 LaTeX 总是尝试以 `h-t-b-p` 的顺序来确定图形的位置。所以 `[hb]` 和 `[bh]` 都会以 `h-b` 的顺序来排版。
- 给出的参数越多，LaTeX 的排版结果就会越好。一般来说，`[htbp]`、`[tbp]`、`[htp]`、`[tp]` 这些组合得到的效果就很不错。
- 只给出单个的参数项 `[t]`、`[b]`、`[p]`、`[h]` 极易引发问题[44]。如果该图形不适合所指定的位置，它就会被搁置并阻碍对接下来图形的处理。一旦这些阻塞的图形数目超过了 18 幅这一 LaTeX 所能容许的最大值，就会产生 “Too Many Unprocessed Floats” 的错误（见第 17.4 节）。

见 [7, 第 198 页]

当 LaTeX “*试图*”放置一浮动图形时，它将遵循以下规则：

1. 图形只能置于由位置参数所确定的地点。
2. 图形的放置不能造成页面溢出（`overfull page`）。
3. 图形只能置于当前页或后面的页中[45]。所以图形只能“向后浮动”而不能“向前浮动”。
4. 图形必须按顺序出现。当前图形只有当之前的图形都被放置好之后才能被放置。进一步又有两个衍生规则：
   - 只要前面有未被处理的图形，当前位置就不会放置图形。
   - 一幅“不可能放置”的图形将阻碍之后的图形放置。直到文件结束或达到 LaTeX 的浮动限制。参见第 17.4 节。

   同样地，表格也只能在其前面的表格都被处理完后才能被放置。不过，表格在排版时是跳过图形而单独处理的。反之，图形的处理也是与表格的处理无关的。
5. 必须符合在第 18 节中给出的审美条件。例如，一页上的浮动体的数目不能超过 `totalnumber`。在浮动位置选项前加上一个惊叹号（如 `\begin{figure}[!ht]`）会使 LaTeX 忽略应用于文本页的审美条件，试图强制放置浮动图形。不过，`!` 不会影响应用于浮动页的审美条件。

[43] 当一幅图形被放置在页面的底部时，如果此页有脚注的话，它将位于所有脚注的下方。现在还没有办法来避免这种情况。

[44] 实际上，永远不要单独使用 `[h]` 选项。由于使用单个的 `[h]` 选项效果过于糟糕，较新版本的 LaTeX 自动将其改为 `[ht]`。

[45] 因为图形可浮动到当前页的顶部，所以它可能会出现在它所在文本的前面。如果要防止出现这种情况，可使用 `flafter` 宏包。只要使用 `\usepackage` 导入宏包即可，不需要使用什么命令。

17.3 清除未处理的浮动图形

使用浮动体的一大优势就是不需要将它们立即放置在输入它们的地方。LaTeX 会将它们暂时保存，直到在更合适的地点加以放置。当一浮动图形已被 LaTeX 读入，但还没有将它放到页面上时，这一图形被称为“未处理浮动图形”。虽然 LaTeX 处理浮动体的算法通常很好，但有时还是需要手动强制处理那些未处理的浮动图形。

下面的三个方法都可以用来清除未处理的浮动图形。这些命令必须分开使用。然而，过度使用这些方法要么说明手动管理浮动体位置过去细致，要么说明位置参数有问题（见第 18 节）。

clearpage

最基本的用来清除未处理的浮动图形的方法就是使用 `\clearpage` 命令。它可让 LaTeX 排版所有未处理的浮动图形并开始一新页。尽管这一命令很有效，但它也常常导致页面的下方出现很大的空白。

FloatBarrier

对于大多数情况，最好的方法是使用 `placeins` 宏包提供的 `\FloatBarrier` 命令。使用 `placeins` 宏包有三种方法：

1. `\FloatBarrier` 使所有未处理的浮动图形立即被处理。但与 `\clearpage` 不同的是，它不会开始一新页。
2. 很普遍的一种情况是要求浮动图形在它们所在的章节中排出，为此可在调用 `placeins` 宏包时使用 `section` 选项：

   ```
   \usepackage[section]{placeins}
   ```

   这样会重新定义 `\section` 命令，在每一节之前都加上一个 `\FloatBarrier` 命令。

   注意这个 `[section]` 选项是很严的。举例来说，如果在一页的中间开始新的一节，那么上面这个 `section` 选项会阻止上一节的浮动图形放置在这一页的底部，因为这样会在新的一节之后。
3. 使用 `below` 选项：

   ```
   \usepackage[below]{placeins}
   ```

   会比使用 `section` 选项松一些。它允许上一节的浮动图形出现在新一节之后，只要在同一页中有上一节的内容。

`placeins` 宏包不会改变浮动体的顺序。例如，由于 `\FloatBarrier` 只能在当前页的顶部放置带 `[t]` 选项的浮动体，因此，`\FloatBarrier` 命令处理时会在当前页留白并另起新一页，然后将带 `[t]` 选项的浮动体放置在新一页的顶端。类似地，`\FloatBarrier` 不能在当前位置放置带 `[b]` 选项的浮动体，因此会禁止文本出现在该浮动体的下面。

这两个例子再次说明了，当指定多个位置选项（例如 `[tbp]` 或 `[htbp]`）时，浮动体位置的处理方式会更加有效。

afterpage/clearpage

`afterpage` 宏包提供了命令 `\afterpage`，可以在下一自然分页时执行一个命令。因此，用

```
\afterpage{\clearpage}
```

会使所有未处理的浮动体在下一分页前被清除完。

使用 `\afterpage{\clearpage}` 并不总可以解决浮动限制问题（见第 17.4 节）。因为它只是在页面结束前才会执行 `\clearpage` 命令，而到页面结束时可能又会累积新的未处理的浮动体，进而可能已超过了 LaTeX 的限制。

`\afterpage{\clearpage}` 命令在排版尺寸较小的浮动页图形时特别有用。而命令 `\floatpagefraction`（见第 18.2 节）会阻止“太小”的图形在浮动页上出现。此外，由于浮动位置标识符 `!` 不会应用于浮动页，因此 `[!p]` 不会破除 `\floatpagefraction` 的限制，使用 `\afterpage{\clearpage}` 既可以克服 `\floatpagefraction` 的限制，又不会导致正文页有较多空白。

17.4 过多未处理的浮动体

如果一浮动体不能被即时处理，它就会被放到未处理的浮动体队列中等待处理。由于这一队列只能有 18 个位置，所以当未处理的浮动体的数目超过这一限制时就会导致发生“Too Many Unprocessed Floats”的错误。造成这种错误有四种可能的原因：

1. 最常见的原因是浮动位置选项与浮动位置参数冲突。例如对于一个带 `[t]` 选项的 `figure` 环境，如果它的高度超过了 `\topfraction` 的值，就会被放到等待处理队列中。其它单位置选项也有类似的问题，所以给出尽可能多的浮动位置选项。
2. 不适当的浮动体比例参数值会造成一些图形无法放置。要防止出现这种情况，一定要确保所使用的比例参数值满足第 18.2 节中对此的要求。
3. 在很少的情况下，如使用了很多浮动体和 `\marginpar` 边注（会和浮动体使用相同的未处理队列机制相同），可能确实需要较长的等待队列。这时可使用 `morefloats` 宏包将等待队列的长度限制增加到 36。
4. 如果超过 18 幅图形在之间没有任何文本的情况下被读入，就会超出 LaTeX 浮动放置队列的最大数目。可能的解决办法有：
   (a) 将图形散布在正文中。这会使得有足够的文本来自然分页，LaTeX 也会更容易地处理浮动体。
   (b) 在这些图形之间加入 `\clearpage`。这样做可能得花费一些时间来调整页面以避免产生有很大空白的页。注意，尽管 `\afterpage{\clearpage}` 会在下一个自然分页处调用 `\clearpage`，但在此情况下并不起作用，因为引发自然分页需要足够的文本，在这之前就会超过浮动体队列限制了。
   (c) 因为这里没有文本，所以图形并不需要浮动。因此，最好的解决办法也许是采用第 21 节中的方法来构建非浮动的图形，同时用 `\vspace` 和 `\vfill` 来提供竖直间距。

18 定制浮动位置

下列的样式参数用于避免页面生成糟糕的浮动体效果，例如一页中的浮动体数量过多或者位置错误。如果在正文中的某处修改这些参数，那么直到下一页才会生效。不过，如

果是在导言区修改的话，就会对整个文档都起作用。

18.1 浮动位置的计数器

表 6 中所给出的三个计数器可用于防止 LaTeX 将过多的浮动体置于同一文本页中，但它们不会影响浮动页。在浮动位置选项前加上 ! 会让 LaTeX 忽略这些计数器。这些计数器的值可用 \setcounter 命令来设置。例如：

```
\setcounter{totalnumber}{2}
```

会阻止 LaTeX 将多于两个的浮动体放置到一文本页中。许多人感觉默认的浮动位置计数器太严格了，更喜欢宽松一些的设置，例如

```
\setcounter{topnumber}{4}
\setcounter{bottomnumber}{4}
\setcounter{totalnumber}{10}
```

表 6: 浮动位置计数器

topnumber	可以位于文本页顶部的浮动体的最大数目（缺省值为 2）。
bottomnumber	可以位于文本页底部的浮动体的最大数目（缺省值为 1）。
totalnumber	可以位于文本页中的浮动体的最大数目（缺省值为 3）。

18.2 图形环境中的比例参数

表 7 中给出的命令用来控制一页中有多大比例的区域可用来放置浮动体（这里的比例是指浮动体的高度除以正文高度 \textheight）。前面三个命令只作用于文本页，而最后一个命令只作用于浮动页。在浮动位置选项前加上 ! 会让 LaTeX 忽略前面三个命令，而 \floatpagefraction 总是起作用的。这些命令的值可以用 \renewcommand 来修改。例如：

```
\renewcommand{\textfraction}{0.3}
```

限定浮动体不得超过文本页的 70%。

比例系数使用指引

这些默认的位置比例参数值既 可以防止过多/过大的浮动体占据文本页面的主要空间，也可以防止在浮动页上有很大的空白而只有很小的图形。虽然这些缺省值让 LaTeX 运作得很好，但有时显得稍稍严了些，结果导致有些浮动体到标明它们的命令距离很远。这种情况下，可以将这些比例的值放宽松些，例如：

```
\renewcommand{\textfraction}{0.15}
\renewcommand{\topfraction}{0.85}
\renewcommand{\bottomfraction}{0.70}
\renewcommand{\floatpagefraction}{0.66}
```

在调整这些比例值的时候必须要小心，不适当的比例值会导致糟糕的排版结果或者大量未处理的浮动体。要避免出现这类问题，应该基于如下的一些原则：

\textfraction

不要让 \textfraction 的值小于 0.15，因为这样的文本页难以阅读。如果一幅图的

高度超过了 \textwidth 的 85%，那么最好将它单独放置到一浮动页上。这样的效果肯定比勉强将它放置到一文本页，而且下方还有一两行文本的效果好得多。

进一步地，永远不要将 \textfraction 的值设为零以便允许文本页中不含文本。这样作会让 LaTeX 感到迷惑并导致低劣的排版结果。

\topfraction

不要使 \topfraction 的值大于 1 − \textfraction，否则会使 LaTeX 的浮动定位算法发生矛盾[46]。

\bottomfraction

好的排版风格不提倡在页面的底部放置太多的图形，故 \bottomfraction 的值一般要比 \topfraction 小。不要使 \bottomfraction 的值大于 1 − \textfraction。否则会使 LaTeX 的浮动定位算法发生矛盾。

\floatpagefraction

如果 \floatpagefraction 的值很小，那么每一浮动页上就只能放置一个浮动体。当放置的浮动体很小的时候，会使浮动页上出现很大面积的空白。

如果 \floatpagefraction 的值大于 \topfraction 的值，使用 [tp] 选项的浮动图形就有可能被阻塞而无法排版。例如，假设一个带有 [tp] 选项的图形高度大于 \topfraction 却小于 \floatpagefraction，那么由于它既无法放置在文本页上，也无法放置在浮动页上，所以就被阻塞而无法处理。为避免出现这样无法处理的情况，\topfraction 和 \floatpagefraction 的值必须满足以下的不等式：

$$\texttt{\floatpagefraction} \leq \texttt{\topfraction} - 0.05.$$

后面的 0.05 这一项是由于文本页和浮动页有不同的竖直间距[47]。同样地，如果使用了 [bp] 或 [hbp]，那么 \floatpagefraction 和 \bottomfraction 要满足：

$$\texttt{\floatpagefraction} \leq \texttt{\bottomfraction} - 0.05.$$

注意缺省值并不满足上面的不等式，因此在处理带有 [bp] 或 [hbp] 选项的图形时偶尔可能会有问题。

18.3 限制浮动

\suppressfloats 阻止在当前页的顶部或底部出现额外的浮动体。但是不会影响带有 h 或者 ! 选项的图形位置。

在一幅图形之前给出 \suppressfloats[t] 会阻止图形出现在文本中位置的上方。进一步地，flafter 宏包重定义了 LaTeX 的浮动算法，可以在整个文档中阻止这种现象。

[46] 旧译本误作“小于 1 − \textfraction”。——译注

[47] 特别地，当比较图形的高度比例和 \topfraction 时，\textfloatsep 和其它文本页浮动间距都被计算在内。而对浮动页来说，在考察图形的高度比例是否超过了 \floatpagefraction 时，浮动间距是不被计算在内的。因此，必须从 \topfraction 中减去 \textfloatsep 除以 \textheight 的值（约为 0.05）。关于图形间距详见第 19.1 节。

表 7: 图形位置的比例参数

\textfraction	页面中必须用于文本的最小比例。缺省值为 0.2，即一页中浮动体所占的比例不得超过 80%。
\topfraction	文本页中页面顶部可以用来放置浮动体的最大空间比例。缺省值为 0.7，即放置在顶部的浮动体所占的高度不得超过整个页面高度 70%。同样地，如果多个使用了选项 t 的浮动体的高度和超过了整个页面高度的 70%，即使它们的数目没有超过 topnumber 的值，仍将一个也不会被放置在页面顶部。
\bottomfraction	文本页中页面底部可以用来放置浮动体的最大空间比例。缺省值为 0.3，即，只有当浮动体的高度不超过整个页面高度的 30% 时才可以放置在页面底部。
\floatpagefraction	浮动页中必须由浮动体占用高度的最小比例。因此在浮动页中空白所占的比例不会超过 1 − \floatpagefraction。缺省值为 0.5。

表 8: Suppressfloats 的选项

\suppressfloats[t]	阻止当前页的顶部出现其它的浮动体。
\suppressfloats[b]	阻止当前页的底部出现其它的浮动体。
\suppressfloats	同时阻止当前页的顶部和底部出现其它的浮动体。

表 9: 文本页面中的浮动图形间距

\floatsep	出现在页面的顶部或底部的浮动体之间的垂直距离。缺省为 12pt plus 2pt minus 2pt。
\textfloatsep	出现在页面的顶部或底部的浮动体与文本之间的垂直距离。缺省为 20pt plus 2pt minus 4pt。
\intextsep	出现在页面中间的浮动体（如使用了 h 选项的浮动体）与上下方文本之间的垂直距离。缺省为 12pt plus 2pt minus 2pt。

表 10: 浮动页面中的浮动体间距

\@fptop	浮动页中顶部的浮动体上方的空白。缺省为 0pt plus 1.0fil。
\@fpsep	浮动页中的浮动体之间的距离。缺省为 8pt plus 2.0fil。
\@fpbot	浮动页中底部的浮动体下方的空白。缺省为 0pt plus 1.0fil。

19 定制 figure 环境

19.1 图形的间距

表 9 中给出的长度控制两幅图形之间或图形与正文的间距。与其它大部分的 LaTeX 长度不同的是，这三个都是橡皮长度，这就使得可以通过缩短或拉伸获得更好的排版页面。这些长度可用 \setlength 命令来设定。例如：

```
\setlength{\floatsep}{10pt plus 3pt minus 2pt}
```

将 \floatsep 的正常值设定为 10 pt。此外，为了改善页面排版，浮动体的间距可以从 8 pt 到 13 pt 之间调节。

由于 LaTeX 在每一个按照选项 h 排在当前位置的浮动体前后都放置一个 \intextsep 间距，因此两个连续的带 h 选项的浮动体之间的间距是 \intextsep 的两倍。为了消除这一额外间距，可以将两个浮动体合并成一个。不过这样的话，浮动位置算法就不会分开放置，得到的排版效果没有分开时那么好。

表 9 中给出的长度不会影响浮动页上各浮动体之间的距离。它们由表 10 中给出的长度控制。浮动页面中的空白经常使用单位 fil，类似于 \vfill 生成的垂直间距，具有无限伸展的能力。当在一段距离中出现多个 fil 时，它们将按比例填充这段距离。例如，默认的浮动页面参数效果是，浮动页面上的浮动体之间的距离是最顶端浮动体到页面顶端距离的两倍，也是最底端浮动体到页面底端距离的两倍。

在表 10 中的长度名字中的 @ 表示这是一个 LaTeX 内部命令[48]。所以，所有改变这些长度的 \setlength 命令都必须放到 \makeatletter 和 \makeatother 之间。例如：

```
\makeatletter
\addtolength{\@fpsep}{4pt}
\makeatother
```

[48] 用户所有涉及到 LaTeX 内部命令的代码都必须包含在 \makeatletter 和 \makeatother 之间。为实现它的功能，LaTeX 使用了很多普通用户无需涉及的内部命令。为防止这些内部命令名字和用户定义的命令的名字发生冲突，LaTeX 在它的内部命令名字前加上了一个 @。由于 LaTeX 命令的名字只能包含字母，所以通常不可能定义一个含有 @ 的命令。不过，命令 \makeatletter 让 LaTeX 把 @ 当作字母，从而可以定义带有 @ 的命令。命令 \makeatother 则重新令 LaTeX 不把 @ 当作字母。

表 11: 图形标尺命令

\topfigrule	该命令在一页顶部的最后一个浮动体后，\textfloatsep 前执行（见第 19.1 节）。
\bottomfigrule	该命令在一页底部第一个浮动体前，\textfloatsep 后执行。

将浮动页中浮动体之间的距离增加了 4pt。

19.2 图形上下方的水平线

通过重新定义 \topfigurerule 和 \bottomfigurerule 命令可在文本和页面顶部或底部的浮动图形之间画一条水平线。尽管 \topfigurerule 和 \bottomfigurerule 是已经定义的 LaTeX 命令，但是由于它们奇特的定义方式，在重定义时要用 \newcommand 而不是 \renewcommand。

为了不破坏版面，这些命令所加标尺的高度必须为零。例如要画一条 0.4pt 的水平线，就必须加上 −0.4pt 的距离：

```
\newcommand{\topfigrule}{\hrule\vspace{-0.4pt}}
```

因为 \topfigrule 在 \textfloatsep 之前被执行，上面的命令没有在图形与水平线之间留出距离。下面的命令则在图形与水平线之间留出了 5pt 的空间。

```
\newcommand{\topfigrule}{%
    \vspace*{5pt}\hrule\vspace{-5.4pt}}
\newcommand{\botfigrule}{%
    \vspace*{-5.4pt}\hrule\vspace{5pt}}
```

在这里 \topfigrule 的定义中，首先向下移动 5pt （进入到 \textfloatsep 的区域）以提供图形与水平线之间的距离，然后画上一高为 0.4pt 的水平线，最后再向上移动 5.4pt 以补偿前面向下的位移。同样地，\botfigrule 在图形与水平线之间留出了 5pt 的空间。

由于上面的命令使得图形与水平线之间的距离为 5pt，所以水平线与文本之间的距离为 \textfloatsep − 5pt（见第 19.1 节）。

水平线的厚度缺省为 0.4pt，并可用 \hrule 命令的 height 选项来改变。

```
\newcommand{\topfigrule}{%
    \vspace*{5pt}{\hrule height0.8pt}\vspace{-5.8pt}}
\newcommand{\botfigrule}{%
    \vspace*{-5.8pt}{\hrule height0.8pt}\vspace{5pt}}
```

需要注意下面几点：

- \topfigrule 和 \botfigrule 命令对浮动页上的图形和放置在当前位置的图形（如使用了 h 选项）不起作用。如果放置在当前位置的图形正好位于页面的顶部或底部，也不会画上水平线。
- 即便图形很宽，水平线的长度也只与文本的宽度相等（参见第 23 节）。
- 因为 LaTeX 的 \rule 命令在 \parskip 不为零时会产生额外的空白，所以代之以 TeX 命令 \hrule。

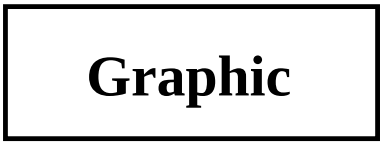
19.3 图形与标题的间距

LaTeX 假定图形的标题位于图形的下方，故而在标题上方保留了更多的空白。[49] 因此

```
\begin{figure}
    \centering
    \caption{图像上方的标题}
    \includegraphics[width=2in]{graphic.eps}
\end{figure}
```

生成的图 15 中标题和图形非常接近。

图 15: 图像上方的标题

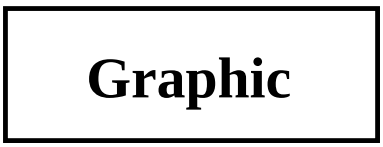


标题的垂直间距由长度 \abovecaptionskip 和 \belowcaptionskip （缺省分别为 10 pt 与零）控制。可以用标准的 LaTeX 命令 \setlength 和 \addtolength 来修改。例如：

```
\begin{figure}
    \setlength{\abovecaptionskip}{0pt}
    \setlength{\belowcaptionskip}{10pt}
    \centering
    \caption{图像上方的标题}
    \includegraphics[width=2in]{graphic.eps}
\end{figure}
```

得到图 16。其中标题的上方没有额外的空白，与图形之间则有 10 pt 的距离。

图 16: 图像上方的标题



如果文档的所有浮动体标题都位于顶部，那么可将如下命令放到导言区：

```
\setlength{\abovecaptionskip}{0pt}
\setlength{\belowcaptionskip}{10pt}
```

从而对整个文档都起作用（包括图形和表格）。如果只是有一部分标题要求位于浮动体的上方，那么可定义如下的命令：

```
\newcommand{\topcaption}{%
    \setlength{\abovecaptionskip}{0pt}%
    \setlength{\belowcaptionskip}{10pt}%
    \caption}
```

[49] 当使用 caption 宏包时，默认选项 position=auto 会自动判断标题的位置并设置合适的间距，因此无需手动调整间距，见第 20 节。——译注

Fig. 17: This is the Caption

此时 \topcaption{标题文本} 生成的标题放在浮动体顶部就很合适。

另外还有两种方法可以使浮动体顶部的标题具有合适的间距：

- 使用 caption 宏包的 position=top 选项（见表 16）可以交换 \abovecaptionskip 和 \belowcaptionskip 的长度。
- topcapt 宏包 [3] 定义了一个 \topcaption 命令，可以在交换 \abovecaptionskip 和 \belowcaptionskip 的长度的同时生成标题。

19.4 标题的标记

缺省情况下，LaTeX 会在图形的标题开头加上像 “Figure 13: ” 这样的标记。其中的 “Figure” 可以通过重定义 \figurename 来更改。例如，下面的命令

```
\begin{figure}
    \centering
    \includegraphics[width=1in]{graphic.eps}
    \renewcommand{\figurename}{Fig.}
    \caption{This is the Caption}
\end{figure}
```

得到如图 17 的结果。至于标题文本的字体，分隔符 “:” 以及其它标题属性的修改可用 caption 宏包（见第 20 节）来完成。

19.5 标题的编号

图形编号的默认方式是使用阿拉伯数字 $1, 2, 3, \ldots$ 这可以通过重新定义 \thefigure 命令来改变。

当前图形的编号存放在 figure 计数器内。而 \thefigure 命令制定用何种计数命令（包括 \arabic、\roman、\Roman、\alph、\Alph 等）打印计数器的值。例如：

```
\renewcommand{\thefigure}{\Roman{figure}}
```

图片编号的效果是大写罗马数字（I, II, III, IV, ...）

关于图片编号的注记：

- 如果使用 \alph 或 \Alph，那么图片的个数不能超过 26 个。
- 由于罗马数字一般比较长（例如比较 18 和 XVIII），因此在图形组成的表格中使用 \Roman 或 \roman 可能导致水平间距问题。

19.6 将图形放于文档的最后

有些期刊要求将图表和正文文本分开排放。这时可用 endfloat 宏包就可以将浮动体放置到文档的最后。直接导入该宏包即可：

```
\usepackage{endfloat}
```

另外，这个宏包在导入时还有一系列选项，包括：

- 在邻近浮动体的文本中会放置像 “[Figure 4 about here.]” 之类的说明。要取消这一功能可在调入宏包时使用 nomarkers 选项。

  ```
  \usepackage[nomarkers]{endfloat}
  ```

  另外，说明中的文本可通过重定义命令 \figureplace 和 \tableplace 来更改。例如：

  ```
  \renewcommand{\figureplace}{%
      \begin{center}%
      [\figurename~\thepostfig\ would appear here.]%
      \end{center}
  ```

  会修改 \figureplace 文本。

- 在图形和表格之前会有一列表。可使用 nofiglist 和 notablist 宏包选项来取消这一功能。

- fighead 和 tabhead 宏包选项分别在图形和表格前加上章节标题。

- 图形放置在表格之前，也可用 tablefirst 宏包选项来改变这一顺序。

- 在每一图形和表格后会执行 \clearpage 命令，从而使得每一页只有一个浮动体。这可通过修改 \efloatseparator 来改变。例如，

  ```
  \renewcommand{\efloatseparator}{\mbox{}}
  ```

  会在每一浮动体后面放置一个空的盒子。

19.7 调整标题行距

在导言区使用

```
\linespread{1.6}
```

或者

```
\renewcommand{\baselinestretch}{1.6}
```

可以实现双倍行距[50]。不过这也会使浮动体标题和脚注也变成双倍行距。为了使双倍行距只应用于正文，而标题和脚注仍是单倍行距，需要使用 setspace 宏包[51]：

```
\usepackage{setspace}
\linestretch{1.5}
```

一个 linstretch 是单倍行距，1.25 倍 linstretch 是 1.5 倍行距，而 1.6 倍 linstretch 是双倍行距。

[50] 这些命令也可以在文档内部使用，以改变某些段落的行间距，不过一般认为这样的设计很糟糕。在文档内部使用时，必须在行距命令之后使用 \normalsize 等字号命令才能使新行距生效。

[51] 尽管 doublespace 宏包也可以设置行距，但是没有很好地更新到 LaTeX 2ε 下，所以会与许多宏包相互影响。取而代之应使用 setspace 宏包。

20 使用 caption 宏包来定制标题

第 19.4 节和第 19.3 节分别介绍了如何定制浮动图形的标题的标签和标题上下方的垂直间距。对于标题其它属性的控制，则可以用 caption 宏包[52] 来完成。本节简要介绍了 caption 宏包，更多细节请参考宏包文档 [16]。

caption 宏包可以用于多种类型的浮动体。它官方支持 float、listings、longtable、rotating、sidecap、supertabular、subfig 等宏包，同时也可以与 floatfig、subfloat、wrapfig 等宏包一起使用[53]。

ccaption 宏包

尽管本文档中没有介绍，不过 ccaption 宏包（注意有两个字母 c）也提供了标题定制的命令，参见 [18]。

20.1 Caption 宏包概述

关于 caption 宏包有几个方面：

- 3.x 版本的 \caption 命令可以生成标题，这将在第 20.2 节介绍并在表 12 列出。
- 第 20.3 节介绍了两种如何定制标题选项的方法。其中选项主要有四种类型：

  字体选项 用于定制标题中的字体[54]。这些选项在表 14 和表 15 列出。相关例子在第 20.4.1 节。

  标题间距选项 控制标题的垂直间距。相关选项在表 16 中列出。相关例子在第 20.4.2 节。

  标题标签选项 用于定制标题的标签和分隔符。相关选项在表 17 中列出。相关例子在第 20.4.3 节。

  标题格式选项 用于定制标题的格式。相关选项在表 18 中列出。相关例子在第 20.4.4 节。

  为了方便查询，关于 caption 宏包选项的表格集中列于 81–84 页。相关例子集中列于第 20.4 节。

- 用户可以定义一组标题选项，称之为**标题样式**。之后可以用 style= 选项指定所有的选项。参见第 20.5.1 节。
- 除了使用内置的选项值外，用户还可以自己定义选项值。参见第 20.5.2 节。

[52] 版本 3.0+ 的 caption 宏包取代了之前版本的 caption 以及过时的 caption2 宏包。

[53] 这份名单有些过时了，例如 floatfig 是一个 LaTeX2.09 宏包，在 LaTeX 2ε 中已扩展为 floatflt。关于最新 3.3 版本的兼容性，其宏包文档 [16] 中的叙述为：

> The caption package was adapted to the following packages which deals with captions, too:
>
> float, floatflt, fltpage, hyperref, hypcap, listings, longtable, picinpar, picins, rotating, setspace, sidecap, subfigure, supertabular, threeparttable, wrapfig, and xtab
>
> Furthermore the floatrow package, the subcaption package (which is part of the caption package bundle), and the subfig package support the caption package and use its \captionsetup interface.

——译注

[54] 尽管 caption 宏包提供了定制标题字体的命令，不过使用的字体不一定包含了所有的字体因子组合。例如，假如用户指定罗马字族、小型大写字样和粗体系列。如果当前字体不支持这一组合，那么 LaTeX 可能会用罗马字族、直立字样和粗体系列的字体来代替。

20.2 标题命令

第 17.1 节和第 19 节分别介绍了 \caption 命令以及相关的定制方法。而 caption 宏包提供了更多的定制化选项。

caption 宏包稍微修改了 \caption 命令，此外还引入了一些新的变形，见表 12。其中的要点包括：

- caption 宏包修改了 \caption 命令，如果可选项指定为空：

  ```
  \caption[]{caption text}
  ```

  那么图列表/表格列表中不会有该标题的条目。

- 新的 \caption* 命令会展示标题内容，但是没有标签编号，也不会出现在列表中。

- 新的 \captionof 命令允许一种特定类型的标题在任何地方使用，包括 figure 环境、table 环境以及文档中的其它地方。例如，

  ```
  \begin{figure}
  ....
  \captionof{table}[List of Tables Text]{Table Caption}
  \end{figure}
  ```

  会在 figure 环境中生成表格标题。这在以下一些情况中会很有用：

  1. 将表格和图形并列排放（见第 30 节）。
  2. 创建边注图形（见第 22 节）。
  3. 创建非浮动图形（见第 21 节）。

  需要注意的是，\captionof 命令应该总是在某个环境内部使用（例如 minipage），这样可以避免在标题和浮动体内容之间分页。

20.3 使用 Caption 宏包命令定制标题

正如第 20.1 提到的，利用 caption 宏包可以定制标题的字体、间距、标签和格式。表 13–18 列出了相应的选项。这些选项可以通过以下两种方式之一来指定。

usepackage 选项 \usepackage[options]{caption}。其中 [options] 可以是表 14–18 中任意选项的组合。例如

```
\usepackage[margin=10pt,font=small,labelfont=bf]{caption}
```

会使得标题两侧缩进了 10 pt，标题的全部（标签和文本）使用 small 字号，并且标签字体加粗。

captionsetup 命令 \captionsetup{options} 命令指定的选项仅在之后的环境中起作用。如果放在导言区，那么 \captionsetup 会作用在整个文档上。例如，

```
\captionsetup{margin=10pt,font=small,labelfont=bf}
```

会使得当前环境中接下来的标题左右缩进 10 pt，整个标题（标签和文本）使用 small 字号，并且标签字体加粗。表 13 介绍了 \captionsetup 和 \clercaptionsetup 命令。

表 12: caption 宏包命令

命令	描述
\caption{⟨*caption text*⟩}	使用 ⟨*caption text*⟩ 作为图表标题以及图表目录的内容。这与不用 caption 宏包时是一样的。
\caption[⟨*list entry*⟩]{⟨*caption text*⟩}	使用 ⟨*caption text*⟩ 作为图表标题，⟨*list entry*⟩ 作为图表目录中的条目。这与不用 caption 宏包时是一样的。
\caption[]{⟨*caption text*⟩}	使用 ⟨*caption text*⟩ 作为图表标题，在图表目录中不创建相应条目。
\caption*{⟨*caption text*⟩}	使用 ⟨*caption text*⟩ 作为图表标题，但标题中不包含标题标签和分隔符（见图 18）。在图表目录中不创建相应条目。
\captionof{⟨*float type*⟩}[⟨*list entry*⟩]{⟨*caption text*⟩}	如果 ⟨*float type*⟩ 是 figure，即使 \captionof 命令位于 figure 环境外，也将生成图标题以及图目录中的条目。类似地，如果 ⟨*float type*⟩ 是 table，即使 \captionof 命令位于 table 环境外，也将生成表格标题以及表格目录中的条目。
\captionof*{float type}{caption text}	类似于 \captionof 命令，⟨*float type*⟩ 指定生成图形还是表格类型标题。类似于 \caption* 命令，\captionof* 命令使用 ⟨*caption text*⟩ 作为标题内容，但不含标题标签和分隔符（见图 18）。在图表目录中不创建相应条目。
\ContinuedFloat	允许多个 \caption 命令使用同一个图形编号。参考第 33.1 节。

表 13: caption 宏包中的标题设置命令

命令	描述
\captionsetup[⟨*float type*⟩]{⟨*options*⟩}	设置标题因子
例 \captionsetup{⟨*options*⟩}	设置所有标题的格式
\captionsetup[figure]{⟨*options*⟩}	设置图形标题的格式
\captionsetup[table]{⟨*options*⟩}	设置表格标题的格式
\clearcaptionsetup{⟨*float type*⟩}	更改标题因子为默认值
例 \clearcaptionsetup{figure}	重置图形标题选项为默认值
\clearcaptionsetup{table}	重置表格标题选项为默认值

表 14: \captionsetup 字体选项

选项	作用部分
font=	作用于整个标题，包括标题标签、分隔符和标题内容
labelfont=	只作用于标题标签和分隔符
textfont=	只作用于标题内容

相比于在 \usepackage 命令中指定可选项，使用 \captionsetup 有两方面的优势。

- \captionsetup 命令中可以指定选项仅作用于图形或仅作用于表格。
- \captionsetup 可以为单独的图形或表格更改设置。例如：

```
\begin{figure}
    ...
    \captionsetup{justification=centering}
    \caption{This is the Caption Text}
\end{figure}
```

会使得该图形的标题居中，但不影响其它图形。

不过，尽管 \captionsetup 可以用于定制单独的标题，但一般来说这是一种糟糕的设计。用户应该在导言区组织 \captionsetup 命令，而避免在文档内使用。

表 15: \captionsetup 字体选项值

作用	选项值	描述
使用所有的字体默认值	default	将字族、字形、字系和字号设置为默认值
设置字族	rm	罗马字族，roman（默认值）
	sf	无衬线字族，sans sarif
	tt	打字机字族，typewriter
设置字形	up	直立字形，upright （默认值）
	it	意大利字形，*italic*
	sl	倾斜字形，*slanted*
	sc	小型大写字形，SMALL CAPS
设置字系	md	中等字系，medium （默认值）
	bf	**粗体字系，bold**
设置字号	scriptsize	scriptsize 字号
	footnotesize	footnotesize 字号
	small	small 字号
	normalsize	normalsize 字号 （默认值）
	large	large 字号
	Large	Large 字号

表 16: \captionsetup 垂直间距选项

关键字	值	描述
skip=	<amount>	设置标题和图表之间的垂直间距，默认是 10 pt。
aboveskip=	<amount>	设置标题和图表之间的垂直间距，默认是 10 pt。正常情况下间距是在标题上方，但是当 position=top 时，aboveskip= 间距则位于与标题下方。
belowskip=	<amount>	设置标题朝图表外侧方向的垂直间距，默认是 0 pt。正常情况下该间距在标题下方，但是当 position=top 时，belowskip= 间距则位于标题上方。
position	bottom, below	将 aboveskip 间距置于标题上方，belowskip 间距置于标题下方。
	top, above	将 aboveskip 间距置于标题上方，belowskip 间距置于标题下方。
	auto	默认值。自动判断标题的位置从而确定间距的方向。在绝大部分情况下没有问题，不过极少数场合可能判断错误。
parskip	<amount>	标题内部的段间距，默认是 0 pt。如果标题只有一个段落，该选项不起作用。

表 17: \captionsetup 标题标签和分隔符选项

关键字	值	描述
labelformat=	default	按照文档类中指定的方式排版标题标签（默认值），一般来说与 simple 效果类似。
	simple	标题标签为标题名和数字。例如“Figure 9”。
	parens	数字放在括号内，例如“Figure (9)”。
	brace	数字以右括号结束，例如“Figure 9)”。
	empty	标题标签为空，没有“Figure”，也没有数字。通常与 labelsep=none 一起使用，以避免分隔符。
labelsep=	colon	标题分隔符为冒号和一个空格（默认值）
	period	标题分隔符为句点和一个空格
	space	标题分隔符为一个空格
	quad	标题分隔符为 \quad
	newline	标题分隔符为 \newline。注意不能与 format=hang 一起使用。
	none	没有标题分隔符。一般只和 labelformat=empty 一起使用。
	endash	标题分隔符为两端加空格的 en dash 连接号“␣--␣”

表 18: \captionsetup 格式选项

关键字	值	描述
format=	plain	标题按正常段落样式排版（默认值）
	hang	标题从第二行起缩进，使得第二行起每行都与第一行的标题文本对齐。
justification=	justified	标题按正常段落样式排版（默认值）
	centerlast	最后一行居中
	centerfirst	第一行居中
	centering	标题每一行都居中对齐
	Centering	与 centering 相同，此外还加入了 TEX 断词算法
	raggedright	每一行靠左对齐
	RaggedRight	与 raggedright 相同，此外还加入了 TEX 断词算法
	raggedleft	每一行靠右对齐
	RaggedLeft	与 raggedleft 相同，此外还加入了 TEX 断词算法
indention=	<amount>	标题第二行起额外的缩进量（默认为 0 pt）
hangindent=	<amount>	标题中每一段第二行起的额外缩进量（默认为 0 pt）。注意 hangindent= 对于每一段的第一行不起作用。如果标题中只有一个段落，那么 hangindent= 和 indention= 是等价的。
margin=	<amount>	左右两端的边距（默认为 0 pt）
width=	<amount>	设置标题的宽度，此时左右两端的边距是相同的。如果同时设置了 margin= 和 width= 选项，最后的选项将起作用。
singlelinecheck=	true, yes, on, 1	如果标题只有一行，那么将保持居中而忽略 justification= 选项（默认值）。
	false, no, off, 0	justification= 格式也作用于单行标题。

20.4 Caption 宏包示例

20.4.1 Caption 宏包字体选项

表 14 列出了 caption 宏包提供的三个定制标题字体的选项。其中“标签”、“分隔符”和“标题内容”的定义如图 18 所示。这三个选项可以修改字族、字形、字系和字号的任意组合，见表 15。

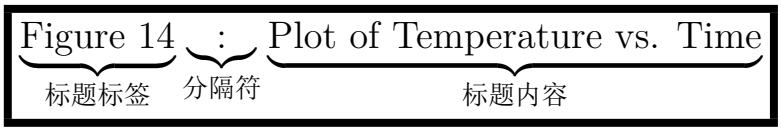


图 18: 标题标签、分隔符和标题内容的定义

字体示例 #1 命令

```
\captionsetup{font={default,Large,bf}}
\caption{This is Caption Font Example \#1}
```

会将所有的标题字体重置为默认值，然后将整个标题（标签、分隔符和标题内容）设置为 Large 字号并加粗，而字族和字形仍然保持默认值，如图 19 所示。

图 19: This is Caption Font Example #1

注意，以上的例子中的代码会同时影响图形和表格的标题。如果 \captionsetup 添加了 [figure] 选项

```
\captionsetup[figure]{font={default,Large,bf}}
```

就会只修改图形标题的字体。

字体示例 #2 命令

```
\captionsetup{font=default, textfont={scriptsize,sf}}
\caption{This is Caption Font Example \#2}
```

会将所有的标题字体重置为默认值，然后将标题内容设置为 scriptsize 字号和无衬线字族，如图 20 所示。

图 20: This is Caption Font Example #2

字体示例 #3 命令

```
\captionsetup{font={default,Large}, labelfont=bf, textfont=sl}
\caption{This is Caption Font Example \#3}
```

会将所有的标题字体重置为默认值，然后将整个标题（标题标签、分隔符和标题内容）设置为 Large 字号，然后将标题标签和分隔符加粗、将标题内容设置为斜体，如图 21 所示。

图 21: *This is Caption Font Example #3*

20.4.2 Caption 垂直间距选项

顾名思义，标题上方的间距由 aboveskip 选项控制，标题下方的间距由 belowskip 选项控制。不过，仅当使用 position=bottom 选项时才成立。当使用 position=top 选项时，aboveskip 和 belowskip 选项的意义是相反的。[55] 遵循如下的原则可以避免发生混淆：

1. 使用 skip= 选项指定标题和浮动体内容的间距。
2. 使用 belowskip= 选项指定标题和周围正文的间距。
3. 默认的选项 position=auto 会自动判断标题的位置。在极少数情况下自动判断失效时，使用 position=bottom 或 position=top 指定标题位于浮动体底部或顶部。

垂直间距示例 #1 使用 skip= 选项修改标题和浮动体内容的间距（默认为 10 pt）。如下命令

```
\captionsetup{skip=1cm}
\caption{Vertical Spacing Example \#1}
```

在标题和浮动体内容之间留出 1 厘米间距，如图 22

图 22: Vertical Spacing Example #1

垂直间距示例 #2 以下例子表明 position=top 会反转 aboveskip 和 belowskip 的意义。

```
\captionsetup{aboveskip=1cm,position=top}
\caption{Vertical Spacing Example \#2}
```

如表 19 所示，标题下方的间距为 1 厘米[56]。

[55] 注意，position=top 只说明标题的上下间距按浮动体顶部的标题来处理，并不意味着标题真的位于图形或表格的上方。标题的实际位置取决于声明 \caption 的位置。——译注

[56] 实际上该设置等价于 \captionsetup{skip=1cm}。这是因为当在 tabular 表格环境或者 \includegraphics 插入命令之前使用 \caption 时，默认选项 position=auto 会自动判断为 top。

表 19: Vertical Spacing Example #2

a	b
c	d

垂直间距示例 #3 一般而言，表格标题位于表格上方，而图形标题位于图形的下方。因此，为方便起见可以分别为表格和图形设置标题位置。

```
\captionsetup{skip=1cm,belowskip=0pt}
\captionsetup[figure]{position=bottom}
\captionsetup[table]{position=top}
...
\caption{Table for Vertical Spacing Example \#3}
\caption{Figure for Vertical Spacing Example \#3}
```

caption 宏包还专门提供了宏包选项 figureposition 和 tableposition 作为以上命令的缩写。上述命令等价于

```
\usepackage[...,figureposition=bottom,tableposition=top]{caption}
\captionsetup{skip=1cm,belowskip=0pt}
...
\caption{Table for Vertical Spacing Example \#3}
\caption{Figure for Vertical Spacing Example \#3}
```

这样表格的标题下方有 1 厘米间距（表 20），而图形的标题上方有 1 厘米间距（图 23）。

表 20: Table for Vertical Spacing Example #3

a	b
c	d

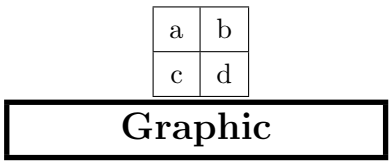


图 23: Figure for Vertical Spacing Example #3

20.4.3 Caption 宏包标签选项

标签选项示例 #1

```
\captionsetup{labelformat=simple}
\caption{This is Caption Label Example \#1}
```

如图 24 所示，标签格式设置为 simple。一般情况下 simple 格式等同于 default 格式，除非文档类重新定义了 default 为不同样式。

图 24: This is Caption Label Example #1

标签选项示例 #2

```
\captionsetup{labelformat=parens}
\caption{This is Caption Label Example \#2}
```

如图 25 所示，标签格式为 `parens`。数字编号位于一对圆括号内。

图 (25): This is Caption Label Example #2

标签选项示例 #3

```
\captionsetup{labelformat=empty, labelsep=none}
\caption{This is Caption Label Example \#3}
```

如图 26 所示，标签格式为 `empty`。不过由于没有编号，很难说这是图 26。

This is Caption Label Example #3

标签选项示例 #4

```
\captionsetup{labelformat=default, labelsep=period}
\caption{This is Caption Label Example \#4}
```

如图 27 所示，标题分隔符由默认的冒号改为句点。

图 27. This is Caption Label Example #4

标签选项示例 #5

```
\captionsetup{labelformat=default, labelsep=newline}
\caption{This is Caption Label Example \#5}
```

如图 28 所示，标题分隔符改成另起一行。

图 28
This is Caption Label Example #5

20.4.4 Caption 宏包格式选项

格式选项示例 #1：标题宽度

```
\captionsetup{width=3in}
\caption{This is an example of customizing the caption width}
```

如图 29 所示，标题宽度设置为 3 英寸。

图 29: This is an example of customizing the caption width

格式选项示例 #2：默认格式 图 30–36 展示了在 `format=default` 格式下，七种不同的 `justification=` 的效果。这些图由如下代码生成：

```
\captionsetup{format=default,justification=justified}
\caption{Caption with default format and justified justification.
Caption with default format and justified justification.
Caption with default format and justified justification.}
...
\captionsetup{format=default,justification=centering}
\caption{Caption with default format and centering justification.
Caption with default format and centering justification.
Caption with default format and centering justification.}
...
\captionsetup{format=default,justification=centerlast}
\caption{Caption with default format and centerlast justification.
Caption with default format and centerlast justification.
Caption with default format and centerlast justification.}
...
\captionsetup{format=default,justification=centerfirst}
\caption{Caption with default format and centerfirst justification.
Caption with default format and centerfirst justification.
Caption with default format and centerfirst justification.}
...
\captionsetup{format=default,justification=raggedright}
\caption{Caption with default format and raggedright justification.
Caption with default format and raggedright justification.
Caption with default format and raggedright justification.}
...
```

```
\captionsetup{format=default,justification=RaggedRight}
\caption{Caption with default format and RaggedRight justification.
Caption with default format and RaggedRight justification.
Caption with default format and RaggedRight justification.}
...
\captionsetup{format=default,justification=raggedleft}
\caption{Caption with default format and raggedleft justification.
Caption with default format and raggedleft justification.
Caption with default format and raggedleft justification.}
```

从图 30–36 中可以看出，标题的第一行与其它行的格式是一样的。

格式选项示例 #3：悬挂模式 图 37–43 展示了在 `format=hang` 格式下，七种不同的 `justification=` 的效果。这些图由如下代码生成：

```
\captionsetup{format=hang,indention=0pt,justification=justified}
\caption{Caption with hang format and justified justification.
Caption with hang format and justified justification.
Caption with hang format and justified justification.}
...
\captionsetup{format=hang,indention=0pt,justification=centering}
\caption{Caption with hang format and centering justification.
Caption with hang format and centering justification.
Caption with hang format and centering justification.}
...
\captionsetup{format=hang,indention=0pt,justification=centerlast}
\caption{Caption with hang format and centerlast justification.
Caption with hang format and centerlast justification.
Caption with hang format and centerlast justification.}
...
\captionsetup{format=hang,indention=0pt,justification=centerfirst}
\caption{Caption with hang format and centerfirst justification.
Caption with hang format and centerfirst justification.
Caption with hang format and centerfirst justification.}
...
\captionsetup{format=hang,indention=0pt,justification=raggedright}
\caption{Caption with hang format and raggedright justification.
Caption with hang format and raggedright justification.
Caption with hang format and raggedright justification.}
...
\captionsetup{format=hang,indention=0pt,justification=RaggedRight}
\caption{Caption with hang format and RaggedRight justification.
Caption with hang format and RaggedRight justification.
Caption with hang format and RaggedRight justification.}
...
\captionsetup{format=hang,indention=0pt,justification=raggedleft}
\caption{Caption with hang format and raggedleft justification.
Caption with hang format and raggedleft justification.
Caption with hang format and raggedleft justification.}
```

图 30: Caption with default format and justified justification. Caption with default format and justified justification. Caption with default format and justified justification.

图 31: Caption with default format and centering justification. Caption with default format and centering justification. Caption with default format and centering justification.

图 32: Caption with default format and centerlast justification. Caption with default format and centerlast justification. Caption with default format and centerlast justification.

图 33: Caption with default format and centerfirst justification. Caption with default format and centerfirst justification. Caption with default format and centerfirst justification.

图 34: Caption with default format and raggedright justification. Caption with default format and raggedright justification. Caption with default format and raggedright justification.

图 35: Caption with default format and RaggedRight justification. Caption with default format and RaggedRight justification. Caption with default format and RaggedRight justification.

图 36: Caption with default format and raggedleft justification. Caption with default format and raggedleft justification. Caption with default format and raggedleft justification.

从图 37–43 中可以看出，format=hang 效果是：标题从第二行有额外的缩进。标题的第一行与其它行的格式是一样的。

20.5 进一步定制

caption 宏包还有一些额外的特性，可以允许用户进行进一步的定制。本节对此进行了简要介绍，详细的指南请参考 [16]。

20.5.1 标题样式

用户可以定义一组标题选项，称之为标题样式。这样通过一个样式选项就可以指定一组选项值。例如，caption 宏包已经定义了名为 default 的选项：

```
\captionsetup{style=default}
```

就等价于

```
\captionsetup{font=default, labelfont=default,
textfont=default, parskip=0pt,
labelformat=simple, labelsep=colon,
format=default, indentation=0pt,
hangindent=0pt, margin=0pt,
parinident=0pt, justification=justified,
singlelinecheck=true}
```

可以在导言区使用 \DeclareCaptionStyle 命令来定义其它标题样式。这些样式既可以是显式地定义所有的参数值，也可以从默认样式出发只修改个别的选项值。例如，在导言区中使用如下 \DeclareCaptionStyle 命令

```
\DeclareCaptionStyle{BigLeft}{style=default, labelsep=period,
font=Large, labelfont=bf,
justification=RaggedRight,
singlelinecheck=false}
```

就可以通过如下命令使用 BigLeft 样式

```
\captionsetup{style=BigLeft}
\caption{This Caption uses BigLeft Style}
```

效果如图 44

图 44. This Caption uses BigLeft Style

20.5.2 其它选项值

caption 宏包提供了一组命令用于其它的选项值，这些命令只能用于导言区中。

```
\DeclareCaptionFont
\DeclareCaptionLabelSeparator
\DeclareCaptionLabelFormat
```

图 37: Caption with hang format and justified justification. Caption with hang format and justified justification. Caption with hang format and justified justification.

图 38: Caption with hang format and justified justification. Caption with hang format and justified justification. Caption with hang format and justified justification.

图 39: Caption with hang format and centerlast justification. Caption with hang format and centerlast justification. Caption with hang format and centerlast justification.

图 40: Caption with hang format and centerfirst justification. Caption with hang format and centerfirst justification. Caption with hang format and centerfirst justification.

图 41: Caption with hang format and raggedright justification. Caption with hang format and raggedright justification. Caption with hang format and raggedright justification.

图 42: Caption with hang format and RaggedRight justification. Caption with hang format and RaggedRight justification. Caption with hang format and RaggedRight justification.

图 43: Caption with hang format and raggedleft justification. Caption with hang format and raggedleft justification. Caption with hang format and raggedleft justification.

```
\DeclareCaptionFormat
\DeclareCaptionJustification
\DeclareCaptionTextFormat
```

选项定义示例 #1 表 15 定义了可以用于 font=、labelfont= 和 textfont= 选项的字体选项值。\DeclareCaptionFont 命令允许用户定义额外的字体选项值。例如，在导言区使用如下命令会定义一个 BigAndBold 选项值

```
\DeclareCaptionFont{BigAndBold}{\Large\bfseries}
```

然后可以使用如下代码

```
\captionsetup{font=BigAndBold}
\caption{This Caption uses a Custom Font}
```

效果如图 45 所示。

Graphic

图 45: This Caption uses a Custom Font

选项定义示例 #2 表 17 描述了如何用 labelformat= 选项控制标题的表头（例如“Figure 33”）部分的格式。\DeclareCaptionLabelFormat 命令允许用户自定义额外的 labelformat= 选项。在自定义中，使用符号 #1 和 #2 指定表头（例如“Figure”）和数字编号的插入地点。例如，在导言区使用如下命令会定义一个 hash 选项值

```
\DeclareCaptionLabelFormat{hash}{#1 {\#}#2}
```

效果是在数字编号之前使用符号 # 。不过，该定义有一个小瑕疵，如果 #1 为空的话，在 #1 后面的空格是不必要的。因此，caption 宏包提供了两个辅助命令。\bothIfFirst 命令有两个变量，只有当第一个变量不为空时才输出这两个变量，否则都不会输出。类似地，\bothIfSecond 仅当第二个变量不为空时才输出这两个变量，否则都不会输出。使用 \bothIfFirst 命令后的定义是

```
\DeclareCaptionLabelFormat{hash}{\bothIfFirst{#1}{ }{\#}#2}
```

该定义要放在导言区，之后可以设置标题格式

```
\captionsetup{labelformat=hash}
\caption{This Caption has a Custom Label Format}
```

效果如图 46。

Graphic

图 #46: This Caption has a Custom Label Format

选项定义示例 #3 表 17 定义了 labelsep= 接受的选项值。此外，用户还可以使用 \DeclareCaptionLabelSeparator 命令为 labelsep= 选项自定义额外的选项值。例如，在导言区使用如下命令会将标签分隔符定义为箭头

```
\DeclareCaptionLabelSeparator{arrow}{\quad\ensuremath{\Rightarrow}\quad}
```

之后如下代码

```
\captionsetup{labelsep=arrow}
\caption{This Caption has a Custom Label Separator}
```

的效果见图 47。

图 47 ⇒ This Caption has a Custom Label Separator

选项定义示例 #4 表 18 介绍了 format= 选项值。\DeclareCaptionFormat 命令允许用户为 format= 选项自定义额外的选项值。在自定义中，使用符号 #1、#2 和 #3 指代标题的各部分。其中 #1 代表标题标签，#2 代表标题分隔符，#3 代表标题文本（其定义可参见图 18）。例如，在导言区使用如下命令定义了 reverse 格式，使得先出现标题内容，然后是标题分隔符，最后是放在双尖括号 \ll 和 \gg 内的标题标签。

```
\DeclareCaptionFormat{reverse}{#3#2\ensuremath{\ll}#1\ensuremath{\gg}}
```

之后如下代码

```
\captionsetup{format=reverse,labelsep=empty}
\caption{This Caption has a Custom Format}
```

的效果见图 48。

This Caption has a Custom Format

≪ 图 48≫

21 不浮动的图形

由于不浮动的图形会生成大段的垂直空白，因此一般认为是比较糟糕的排版风格。此时强烈建议用户使用 figure 环境的 [!ht] 选项，这样只有当当前页面排不下时才会移动图形。

如同第 17 节所介绍的那样，LaTeX 允许“浮动”的图形和表格以增强排版效果。不过，偶尔也会希望一幅图形不要浮动，就放置在与它在 LaTeX 源文件中相同的位置。虽然 \caption 命令只能在 figure 和 table 环境中使用，但是 caption 宏包定义了 \captionof 命令，该命令有两个选项：标题的类型（表格、图形等）以及标题的内容，这样就可以在 figure 和 table 环境之外使用了。使用命令

```
\captionof{figure}{caption text}
```

就可以创建一个图形标题，而不用关心是否真的出现在 figure 环境中。类似地，使用命令

```
\captionof{table}{caption text}
```

会创建一个表格标题。而如下的命令

```
This is the text before the figure.
\\[\intextsep]
\begin{minipage}{\linewidth}
    \centering
    \includegraphics[width=2in]{graphic}%
    \captionof{figure}{This is a non-floating figure}
    \label{fig:non:float}
\end{minipage}
\\[\intextsep]
This is the text after the figure.
```

可得到一幅不浮动的图形。对于不浮动的图形，需要注意下面几点：

- 需要使用 minipage 小页环境来防止在图形中出现分页的情况。
- 图形前后的命令 \\[\intextsep] 作用是进行换行并加上垂直的间距。\intextsep 命令（见第 19.1 节）可以使用任意大小的空白，目的是使不浮动的图形具有与浮动图形相同的上下间距。
- 正常情况下，浮动图形是按照它们在 LaTeX 源文件中的顺序一一放置的。而不浮动的图形是被立即放置到页面上，所以可以跳过浮动图形队列中的未处理图形。如果出现这样情况，那么图形不会按照数字顺序编号[57]。要避免这种顺序错乱，可在不浮动的图形前用 \clearpage 或 \FloatBarrier 命令清除未处理的浮动图形（见第 17.3 节）。
- \captionof 命令也可以用于生成边注图形（见第 22 节）以及与图形并列的表格（见第 30 节）。

21.1 不使用 caption 宏包的非浮动图形

以上篇幅介绍了如何使用 caption 宏包的 \captionof 命令来创建 figure/table 环境之外的标题。本节介绍如何不使用 caption 宏包做到这一点。

\caption 命令之所以可以在 figure 和 table 环境中使用，是因为这两个环境分别为图形和表格定义了内部命令 \@captype。这样，通过定义 \@captype 就可以在 figure 和 table 环境外使用 \caption 命令。当然这时 \@captype 必须用 \makeatletter-\makeatother 命令对包围起来，使得可以在命令名中使用 @。当然，可以每次都使用如下的命令：

```
\includegraphics{file}
\makeatletter\def\@captype{figure}\makeatother
\caption{This is the caption}
```

不过，更方便的做法是在导言区中定义下面的命令：

[57] 在这种情况下，图形目录中图形的顺序是按照图形出现的顺序，而不是图形编号的顺序。

```
\makeatletter
\newcommand\figcaption{\def\@captype{figure}\caption}
\newcommand\tabcaption{\def\@captype{table}\caption}
\makeatother
```

这样定义了两个命令 \figcaption 和 \tabcaption。在正文中无论是否在图形环境中，都可用 \figcaption 得到图形标题。同样地，无论是否在表格环境中，都可用 \tabcaption 得到表格标题。

21.2 float 宏包中的 [H] 位置选项

float 宏包[58] 为 figure 环境加上了一个 [H] 位置选项，从而使得用 figure 环境可以生成不浮动的图形。如下代码

```
\usepackage{float}
...
\begin{figure}[H]
    .....
\end{figure}
```

可以生成不浮动的图形。

如果当前页没有足够的空间放置一幅使用了 [H] 位置选项的图形，该图形会被置于下一页的顶部。如果当前页中有脚注的话，它将会紧接在文本后排出，而不是像通常那样置于页面的底部。如果不想要这种效果，可以使用第 21 节介绍的 \captionof 命令来代替 float 宏包的 [H] 位置选项。

22 边注图形

\marginpar 命令可以用来生成边注。除非使用了 \reversemarginpar 命令（本文档就用了该命令），边注一般放在页面的右边（在 twoside 格式的文档中放在页面的外侧）。边注的宽度由长度 \marginparwidth 控制，而与正文之间的水平距离由 \marginparsep 决定。

边注的第一行与包含 \marginpar 命令的正文文本的那一行对齐（边注的第一行的参考点与当前基线对齐）。

边注不能分页，如果一个边注靠近页面的底部而无法正常排下时，它会伸入页面的底边空白继续排出。如果前面一个边注干扰了后面的边注，那么 LaTeX 会把后面的边注向下移动，但不会移到下一页。所以在最后完成排版前可能要调整一下边注的位置以防它离分页的地方太近。

由于 figure 环境不能在边注中使用，所以无法直接得到浮动的边注图形。这时，可以用第 21 节定义的 \captionof 命令来构造非浮动的边注图形。例如，图 49 就由下面的命令来得到：

Graphic

图 49: 这是边注图形

```
... 构造非浮动的边注图形。
\marginpar{\centering
    \includegraphics[width=\marginparwidth]{graphic.eps}%
```

[58] float 宏包允许用户新的浮动体，如 Program、Algorithm 等。也可以定制加框的和加线条的浮动式样。这些可选的浮动式样重定义了 \caption 命令，使得无论将 \caption 命令放在何处，总会在特定的地方排版标题，因此不可以创建并列图形以及其它的复杂图形。

```
    \captionsetup{type=figure}
    \caption{This is a Marginal Figure}
    \label{fig:marginal:fig} }
例如，图~\ref{fig:marginal:fig} 就由下面的命令来得到：
```

图 49 中图像底部与 \marginpar 命令所在文本的基线是对齐的。对于使用边注图形，需要注意的是：

- 由于边注图形都比较窄小，所以在 \caption 命令之前使用 caption 宏包命令

  ```
  \captionstyle{justification=raggedright}
  \captionstyle{justification=raggedleft}
  ```

  可能会得到更好的标题格式效果。此外，caption 宏包命令

  ```
  \captionsetup{font=small}
  ```

  可以减小标题的字体。关于 caption 的信息详见第 20 节。

- 如同第 21 节所介绍的非浮动图形，边注图形会在未处理的浮动图形前排出。因此，如果希望图形按顺序出现的话，必须在边注之前使用 \clearpage 或 \FloatBarrier 命令。

- 边注的处理机制和浮动图表的处理机制是一样的，所以如果使用了太多的浮动图表和边注，就可能超出 LaTeX 所允许的未处理的浮动体的数目。这时使用 morefloat 宏包是一种解决办法。具体见第 17.4 节。

23 宽图形的处理

排版的易读性原则限制了一行文本中的字符个数。如果不是使用大字体或双列版式，这样的易读性原则就会使得页面的页边距很大，特别当使用 8.5×11 英寸的 letter 页面时更是如此。在第 22 节中展示了边空可以用来放置边注图形。另外一种方法则是创建一个常规的浮动图形，并且伸到一边或者两边的边空中。这可通过在浮动图形环境中嵌套一个很宽的列表环境来实现。例如，可以在导言区加入下列代码来定义一个 narrow 环境：

```
\newenvironment{narrow}[2]{%
    \begin{list}{}{%
        \setlength{\topsep}{0pt}%
        \setlength{\leftmargin}{#1}%
        \setlength{\rightmargin}{#2}%
        \setlength{\listparindent}{\parindent}%
        \setlength{\itemindent}{\parindent}%
        \setlength{\parsep}{\parskip}}%
    \item[]}{\end{list}}
```

那么，所有位于 \begin[narrow][1in]{2in} 和 \end{narrow} 之间的文本都被向左缩进 1 英寸，向右缩进 2 英寸。当使用负长度时，文本就会延伸到边空上去。

图 50: 这是宽图形

23.1 单面版式中的宽图形

在使用单面版式排版时，页面左右的边空不会因奇偶页而取不同的值，故可以不用考虑图形浮动到奇数页或偶数页的问题。下面的命令利用前面定义的 narrow 环境使得图形左边延伸到左边空中 1 英寸，见图 50。

```
\begin{figure}
    \begin{narrow}{-1in}{0in}
        \includegraphics[width=\linewidth]{wide}
        \caption{这是宽图形}
    \end{narrow}
\end{figure}
```

这里给定宽度参数为 \linewidth 使得图形的宽度和 narrow 环境的宽度相等。若给定宽度参数为 \textwidth 会使图形的宽度和原来的正文宽度一样。

当使用边注时，可能希望宽图形能够精确地延伸到边注的边界（使得图形的宽度为 \linewidth + \marginparwidth + \marginparsep）。这时可以定义新长度 \marginwidth 并将它设为 \marginparwidth + \marginparsep。例如：

```
\newlength{\marginwidth}
\setlength{\marginwidth}{\marginparwidth}
\addtolength{\marginwidth}{\marginparsep}
```

接着在 \begin{narrow} 的选项中使用 -\marginwidth 来达到目的。

23.2 双面版式中的宽图形

在使用双面版式排版时，页面左右的边空因奇偶页而取不同的值，且使用宽图形时常常希望图形延伸到装订的那一边（奇数页的左边，偶数页的右边）。在这种情形下，需要使用 ifthen 宏包提供的 \ifthenelse 命令，进而根据图形出现在奇数页或偶数页而选取不同的命令。例如：

```
\usepackage{ifthen}
...
\begin{figure}
    \ifthenelse{\isodd{\pageref{fig:wide}}}%
    {% BEGIN ODD-PAGE FIGURE
        \begin{narrow}{0in}{-1in}
            \includegraphics[width=\linewidth]{file}
            \caption{Figure Caption}
            \label{fig:wide}
        \end{narrow}
    }% END ODD-PAGE FIGURE
```

```
    {% BEGIN EVEN-PAGE FIGURE
        \begin{narrow}{-1in}{0in}
            \includegraphics[width=\linewidth]{file}
            \caption{Figure Caption}
            \label{fig:wide}
        \end{narrow}
    }% END EVEN-PAGE FIGURE
\end{figure}
```

由于 `\ifthenelse` 使用命令 `\pageref` 作为输入，所以需要 LaTeX 运行足够的次数后才能正确地排版交叉引用。

24 横排的图形

在竖排版的文档中，有三种方法可以得到横排的图形。

1. `lscape` 宏包提供了一个 `landscape` 环境，将纸张的左边界作为页面的顶部，使得在此环境中的文本，表格和图形都被横排。
2. `rotating` 宏包提供了一个 `sidewaysfigure` 环境，与 `figure` 环境相似，只是其中的图形被横排。
3. `rotating` 宏包提供了一个 `\rotcaption` 命令，与 `\caption` 命令相似，只是标题被横排。

以上三种方法的区别：

- 方法 1 和 2 将横排的图形放到单独的一页上，而方法 3 则生成独立的浮动体，并不需要单独一页来放置。
- 方法 2 只是将其中的图形横排，而方法 1 中的 `landscape` 环境是一种通用环境，可以横排任何文本、图形和表格。`landscape` 环境还具有分页的能力，可连续生成多个横排页面[59]。
- 使用方法 2 得到的整页图形可以浮动以达到更好的排版效果，而方法 1 得到的图形是不能浮动的[60]。
- 因为方法 1 和 3 使用 `figure` 环境，所以它们可以和 `endfloat` 宏包（见第 19.6 节）一起使用。
- 方法 1 和 2 特别适用于并列的横排图像（关于并列图形的方法见第 28 节）。

24.1 Landscape 环境

`lscape` 宏包是 LaTeX 标准图形宏集的一部分，它定义的 `landscape` 环境可以将横排页面放置在竖排版的文档中。横排页被旋转使得竖排页的左边界为其顶部。

[59] `landscape` 环境能很好地与 `longtable` 宏包配合使用，从而得到连续多页横排的超长表格。

[60] 在 `landscape` 环境中声明的浮动图形只能在该横排页内浮动。

输入命令 \begin{landscape} 会立即排版所有未处理的竖排浮动体，然后进入横排方向页面。同样地，输入命令 \end{landscape} 会立即排版所有未处理的横排浮动体，然后重新回到竖排状态。

所有位于 landscape 环境中的内容都会被横排，例如任何文本、图形和表格的组合。如果只有包含一个浮动图形环境

```
\begin{landscape}
    \begin{figure}
        \centering
        \includegraphics[width=4in]{graphic}
        \caption{Landscape Figure}
    \end{figure}
\end{landscape}
```

这是会得到一个横排的图形。不过，由于 landscape 开始一新页，可能会导致页面出现很大空白，如同本页一样。

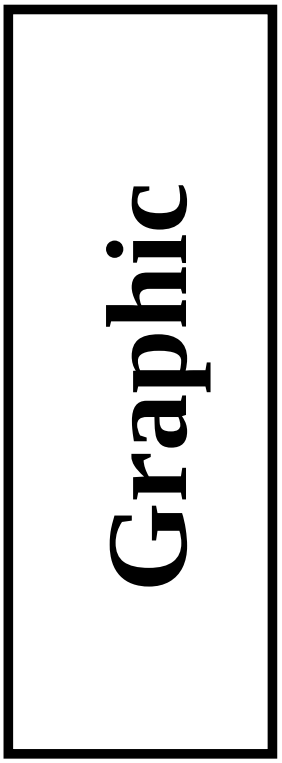


图 51: Landscape Figure

24.2 Sidewaysfigure 环境

rotating 宏包提供了 sidewaysfigure 环境生成横排的图形[61]。例如：

```
\begin{sidewaysfigure}
    \centering
    \includegraphics[width=4in]{graphic}
    \caption{Sidewaysfigure Figure}
\end{sidewaysfigure}
```

得到图 52。

与 landscape 环境不同的是，由 sidewaysfigure 得到的图形可在竖排页中浮动，从而避免导致出现过多空白的页面。不过 landscape 环境则有更大的灵活性，允许横排页中有文本，表格和图形等。

sidewaysfigure 生成图形的默认方向取决于该文档使用 oneside 还是 twoside 文档类选项。

- 当使用 oneside 时，图形的底部面向竖排页的右边。
- 当使用 twoside 时，图形的底部面向竖排页的外边界。

使用 rotating 的宏包选项可以覆盖默认设置。例如：

```
\usepackage[figuresleft]{rotating}
```

使得用 sidewaysfigure 图像的底部位于竖排页的左边（无论是 oneside 还是 twoside）。同样地，

```
\usepackage[figuresright]{rotating}
```

使得用 sidewaysfigure 图像的底部面向竖排页的右边界。

24.3 Rotcaption 命令

用第 24.1 节和第 24.2 节的方法得到的横排图形都是放在一单独的横排页上的。不过对于比较小的图形来说显然没有必要。这种情况下，可以利用 rotating 宏包中的 \rotcaption 来得到小的横排图形。例如：

```
\begin{figure}
    \centering
    \begin{minipage}[c]{1in}
        \hfill\includegraphics[width=2in,angle=90]{graphic}
    \end{minipage}%
    \hspace{0.2in}%
    \begin{minipage}[c]{0.5in}
        \captionsetup{width=2in}
        \rotcaption{由 Rotcaption 命令创建的标题}
        \label{fig:rotcaption}
    \end{minipage}
\end{figure}
```

[61] rotating 宏包还提供了一个 sidewaystable 环境，可以生成横排的表格。

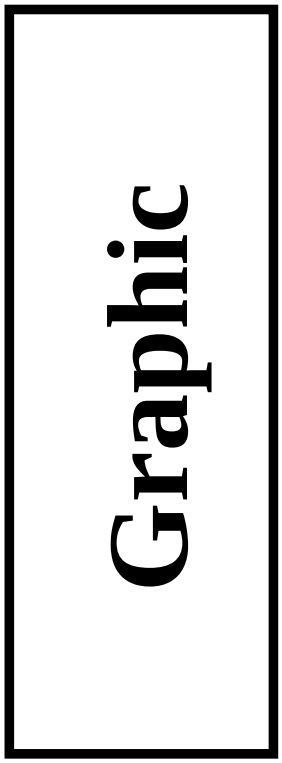


图 52: Sidewaysfigure Figure

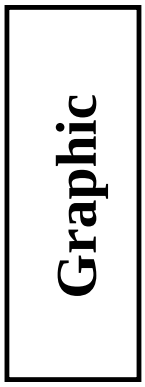


图 53: 由 Rotcaption 命令创建的标题

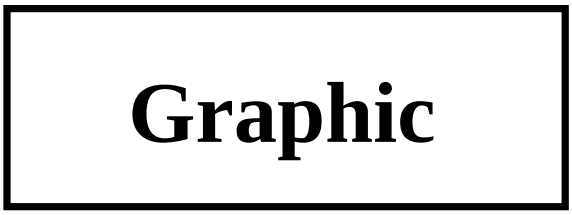


图 54: This is a SCfigure

会生成图 53。\rotcaption 命令生成标题的底部总是面向页面的右边界。与第 24.1 节和第 24.2 节不同的是，\rotcaption 并不旋转图形。因此上例中的 \includegraphics 命令需要使用 angle=90 这一选项。

25 标题在一侧的图形

一般来说，图形的标题放置在其上方或下方。本节将介绍怎样将标题放置在图形的一侧[62]。

25.1 Sidecap 宏包

创建侧边标题最简单的途径就是使用 sidecap 宏包。该宏包定义了 SCfigure 和 SCtable 环境。在其中使用 \caption 命令时，标题就会自动放在环境内容的一侧。例如：

```
\usepackage{sidecap}
...
\begin{SCfigure}
    \includegraphics[width=3in]{graphic}
    \caption{This is a SCfigure}
\end{SCfigure}
```

会生成图 54。

sidecap 宏包在用 \usepackage 调入时有下面四个可选项：

outercaption 标题在偶数页中出现在左侧，奇数页中出现在右侧。这也是 sidecap 宏包的缺省选项。

innercaption 标题在偶数页中出现在右侧，奇数页中出现在左侧。

leftcaption 标题总出现在左侧。

[62] 因为 float 宏包定义的 figure 环境将标题固定在图形的下方，因此无法使用它来得到置于图形一侧的标题。不过，只要没有声明 \restylefloat 命令，其它的 float 宏包的命令都可使用。

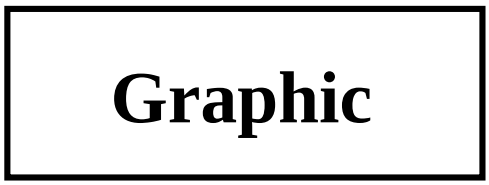


图 55: Caption on the Side

rightcaption 标题总出现在右侧。

Scfigure 环境包括下面两个可选参数：

- 第一个可选参数指定标题对于图形的相对宽度。比较大的值（如 100）会让标题使用最大可能的宽度。缺省为 1。
- 第二个可选参数指定图形的浮动位置选项，如 [htp] 或 [!ht] 等，详见第 17.2 节。

25.2 不使用 Sidecap 宏包时的一侧标题

如果 sidecap 没有提供足够的灵活性，用户还可以使用本节的方法生成位于一侧的标题。第 25.2.1 节展示了如何将标题放在图像的左侧。放在图像右侧的方法也是类似的。第 25.2.2 节展示了对于双面版式的文档，如何将标题置于图形内侧（奇数页中为图形的左侧，偶数页中为图形的右侧）。

25.2.1 图形左侧标题

\caption 命令一般将标题置于图形或表格的下方。可以利用小页环境的技巧使得 \caption 命令把标题放在图形的一侧。例如命令：

```
\begin{figure}
    \centering
    \begin{minipage}[c]{.45\linewidth}
        \centering
        \caption{Caption on the Side}
        \label{fig:side:caption}
    \end{minipage}%
    \begin{minipage}[c]{.45\linewidth}
        \centering
        \includegraphics[width=\linewidth]{graphic}
    \end{minipage}
\end{figure}
```

得到图 55。在小页之间加入像 \hfill 或 \hspace{.05\linewidth} 的水平距离可能会更好些。

图 55 中标题和图形垂直居中。如果想让图形和标题顶部对齐或底部对齐，可参见第 11.4 节。

25.2.2 图形内侧标题

上节图 55 中将标题放在图形的左侧，而对于双面版式的文档，常常会希望将标题置于图形的内侧。这时可用 ifthen 宏包的 \ifthenelse 命令来指定对奇数页和偶数页所使用的不同代码。例如：

```
\usepackage{ifthen}
...
\begin{figure}
    \centering
    \ifthenelse{\isodd{\pageref{fig:side:caption}}}
    {% BEGIN ODD-PAGE FIGURE
        \begin{minipage}[c]{.45\linewidth}
            \centering
            \caption{Caption on the Side}
            \label{fig:side:caption}
        \end{minipage}%
        \hspace{0.05\linewidth}%
        \begin{minipage}[c]{.45\linewidth}
            \includegraphics[width=\linewidth]{graphic}
        \end{minipage}%
    }% END ODD-PAGE FIGURE
    {% BEGIN EVEN-PAGE FIGURE
        \begin{minipage}[c]{.45\linewidth}
            \includegraphics[width=\linewidth]{graphic}
        \end{minipage}%
        \hspace{0.05\linewidth}%
        \begin{minipage}[c]{.45\linewidth}
            \centering
            \caption{Caption on the Side}
            \label{fig:side:caption}
        \end{minipage}%
    }% END EVEN-PAGE FIGURE
\end{figure}
```

这样生成的图形其标题总在图形的内侧。

26 奇偶页中的图形

图形环境的浮动放置算法不能控制图形出现在奇数页还是偶数页。要达到控制浮动图形的奇数或偶数页放置，必须使用 afterpage 宏包的 \afterpage 命令和 ifthen 宏包的 \ifthenelse 命令。

创建图形的一般方法是将图像放置于 figure 环境中。然而，由于 figure 环境是浮动的，因此不能保证要求放置在偶数页的图形不会浮动到奇数页（或者要求在奇数页的图形浮动到偶数页）。

不过，第 21 节介绍的 \captionof 命令可以在不使用 figure 环境的条件下创建一个图形。然后用 \ifthenelse 命令将出现在奇数页上的图形放到下一偶数页上。这需要重复使用两次插图命令，分别对应下一页是奇数和下一页是偶数的情形。为了简化代码，定义 \leftfig 命令如下：

```
\newcommand\leftfig{%
    \vspace*{\fill}%
    \centering
    \includegraphics{graphic}
    \captionof{figure}{This is on the left (even) page.}
```

```
\vspace*{\fill}\newpage}
```

接下来使用这个新定义的命令以及 \afterpage、\ifthenelse 就可以生成一幅只出现左页上的图形：

```
\afterpage{\clearpage%
    \ifthenelse{\isodd{\value{page}}}%
    {\afterpage{\leftfig}}%
    {\leftfig}}
```

关于奇偶页插图的几点说明：

- 欲使图形只出现在右页（奇数页）上，掉换一下 \ifthenelse 的参数顺序即可。

```
\afterpage{\clearpage%
    \ifthenelse{\isodd{\value{page}}}%
    {\leftfig}}%
{\afterpage{\leftfig}}
```

- 由于这些图形是非浮动的，因此可以用 \value{page} 命令确定当前页码。注意，\value{page} 对于浮动图形用处不大，因为它指的是图形处理时的页码，不是图形放置的页码。因此，使用 \value{page} 比 \pageref 更好，因为 \pageref 只有在 LaTeX 的交叉引用收敛时才正确。

- 当图形较大时，可能会出现在图形中间（例如图形与标题之间）分页的情况。这时可将它放到一个小页环境中以保持它的完整性。

```
\newcommand\leftfig{%
    \vspace*{\fill}%
    \begin{minipage}{\linewidth}
        \centering
        \includegraphics{graphic}
        \captionof{figure}{This is on the left (even) page.}
    \end{minipage}
    \vspace*{\fill}\newpage}
```

- \afterpage 命令在极少数情况下会造成 “lost float” 的错误。这时将 \clearpage 从 \ifthenelse 前去掉可能会有所帮助。

```
\afterpage{\ifthenelse{\isodd{\value{page}}}%
    {\afterpage{\leftfig}}%
    {\leftfig}}
```

- 在上面的例子中，图形是占据完整的偶数页。要将其置于偶数页的顶部，修改或去掉 \vspace*{\fill} 和 \newpage 命令：

```
\newcommand\leftfig{%
    \centering
    \includegraphics{graphic}
    \captionof{figure}{This is at the top of the left (even) page.}
    \vspace{\floatsep}}
```

26.1 迎面页图形

在双面版式的文档中，为了方便图形的比较，常常希望将两幅图形分别放在相对的迎面页（facing page）上。为此，需要使用类似于之前一节中放置奇偶页图形的方法。为简单起见，定义命令 \facingfigures 如下：

```
\newcommand\facingfigures{%
    \vspace*{\fill}%
    \centering
    \includegraphics{left}
    \captionof{figure}{This is on the left (even) page.}
    \vspace*{\fill}\newpage\vspace*{\fill}%
    \centering
    \includegraphics{right}
    \captionof{figure}{This is on the right (odd) page.}
    \vspace*{\fill}\newpage}
```

这时可用 \facingfigures 与 \afterpage、\ifthenelse 一起生成迎面页图形：

```
\afterpage{\clearpage%
    \ifthenelse{\isodd{\value{page}}}%
    {\afterpage{\facingfigures}}%
    {\facingfigures}}
```

27 盒子中的图形

盒子中的图形通常指下面两种情形：

- 图形在盒子中，但其标题在盒子之外。
- 图形及其标题都在盒子中。

将某一对象置于盒子中的最基本的方法就是把它放到 \fbox 命令中，这样会将该对象用一长方形的框围起来。fancybox 宏包提供了更多不同式样的盒子。

27.1 图形在盒子中

把 \includegraphics 命令放到 \fbox 中会使所插入的图形置于一个带框盒子中。例如：

```
\begin{figure}
    \centering
    \fbox{\includegraphics[totalheight=2in]{file}}
    \caption{图形在盒子中，但标题在盒子外}
    \label{fig:boxed_graphic}
\end{figure}
```

如图 56 所示，图形被置于一带框盒子中。

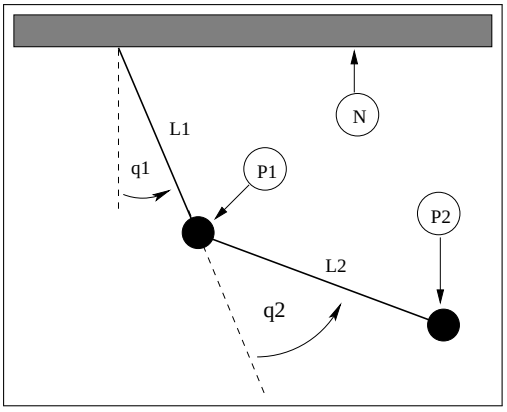


图 56: 图形在盒子中，但标题在盒子外

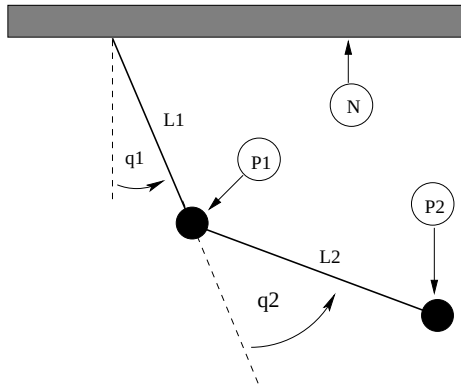


图 57: 图形和标题都在盒子中

27.2 图形与标题均在盒子中

要将图形与标题均置于盒子中，也许有人想当然的以为把 \caption 命令也放到 \fbox 命中就可以了。然而，这样做是无效的，因为 \caption 命令只能在段落模式中使用，而 \fbox 命令中的内容是在左右模式中被处理[63]。

因为小页环境和 \parbox 命令的内容都在段落模式中处理，所以将 \fbox 命令的内容放到小页环境或 \parbox 命令中，就可以把 \caption 包含在 \fbox 中。然而，由于小页环境和 \parbox 命令都必须给出它们的宽度，故没有直接的办法让 \fbox 和图形及其标题一样宽。例如下列命令：

```
\begin{figure}
    \centering
    \fbox{ \begin{minipage}{4 in}
            \centering
            \includegraphics[totalheight=2in]{pend}
            \caption{图形和标题都在盒子中}
            \label{fig:boxed_figure}
        \end{minipage} }
\end{figure}
```

得到图 57，其中图形与标题都置于盒子中。

[63] LaTeX 使用三种模式，左右模式，段落模式和数学模式。参考 [7, 第 36 页]

一般而言，需要通过不断的尝试修改才能来确定小页环境的宽度，从而使得盒子能够恰好围住图形和标题。不过下面的这些方法可以避免枯燥麻烦的尝试修改。

1. 任选一个确定的小页宽度，设置图形的宽度与其相同。

```
\includegraphics[width=\textwidth]{pend}
```

2. 当指定图形的高度时，将图像放在盒子中，然后测量盒子的高度，即可计算出合适的小页宽度。

```
\newsavebox{\mybox}
\newlength{\mylength}
\sbox{\mybox}{\includegraphics[height=3in]{file}}
\settowidth{\mylength}{\usebox{\mybox}}
\begin{figure}
    \centering
    \fbox{ \begin{minipage}{\mylength}
            \centering
            \usebox{\mybox}
            \caption{图形和标题都在盒子中}
            \label{fig:boxed_figure}
        \end{minipage} }
\end{figure}
```

3. 为保证标题只有一行，可以使用 \settowidth 命令来估计标题的宽度并将其作为小页的宽度。

```
\newlength{\mylength}
\settowidth{\mylength}{Figure X: Box Around Figure Graphic and Caption}
\fbox{ \begin{minipage}{\mylength}
...
```

27.3 定制 fbox 的参数

在图 56 和 57 中，盒子的框线厚度为 0.4 pt，在框线和图形之间有 3 pt 的空白。可以使用 \setlength 命令设置 LaTeX 长度变量 \fboxrule 和 \fboxsep，进而修改这些长度。例如命令：

```
\begin{figure}
    \centering
    \setlength{\fboxrule}{3pt}
    \setlength{\fboxsep}{1cm}
    \fbox{\includegraphics[totalheight=2in]{pend}}
    \caption{带有定制盒子的图形}
    \label{fig:boxed_custom}
\end{figure}
```

使得盒子的边框线厚为 3 pt 且其与图形间的距离为 1 厘米。如图 58 所示。

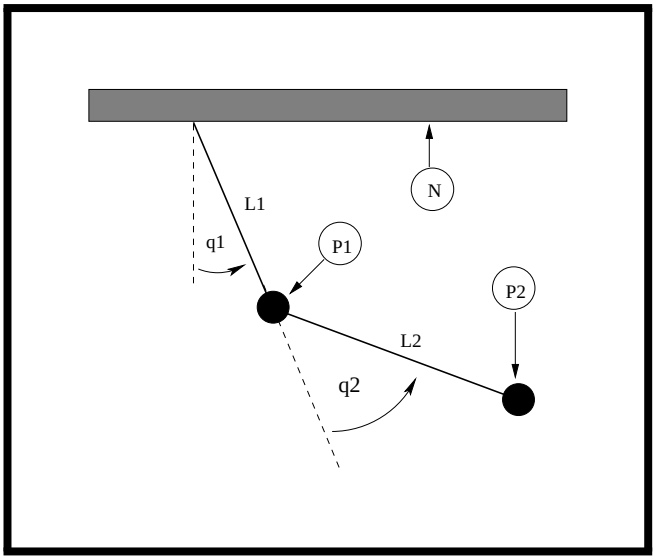


图 58: 带有定制盒子的图形

27.4 fancybox 宏包

在图 56、57 和 58 中，使用 \fbox 命令将图形包围在标准的长方形框盒子中。要想使用不同类型的盒子，可使用 fancybox 宏包。它提供了 \shadowbox、\doublebox、\ovalbox 和 \Ovalbox 等命令来生成不同形状的盒子，如表 21 所示。

如同 \fbox 命令一样，这些盒子命令中的内容与边框间距由 LaTeX 长度 \fboxsep 控制。与第 27.3 节中的 \fboxrule 和 \fboxsep 相同，长度 \shadowsize 也可用 \setlength 命令来设定。而 \ovalbox 和 \Ovalbox 命令中的边框线厚度对应于 picture 环境中的 \thinlines 和 \thicklines 的值，由于它们不是长度变量，所以无法用 \setlength 来设定。这两个宏的值依赖于当前字体的大小和形状，缺省分别为 0.4 pt 和 0.8 pt。例如：

```
\begin{figure}
    \centering
    \shadowbox{ \begin{minipage}{3.5 in}
            \centering
            \includegraphics[totalheight=2in]{pend}
            \caption{环绕整个图形的阴影框}
            \label{fig:boxed_fancy}
    \end{minipage} }
\end{figure}
```

用一个带阴影的盒子将图形与标题包围起来，如图 59 所示。

表 21: fancybox 命令

命令	参数
\shadowbox{Example} Example	• 盒子边框线厚度为 \fboxrule • 盒子阴影厚度为 \shadowsize （缺省为 4 pt ）。
\doublebox{Example} Example	• 内框线厚为 .75\fboxrule。 • 外框线厚为 1.5\fboxrule。 • 内外框之间的距离为 1.5\fboxrule + 0.5 pt。
\ovalbox{Example} Example	• 盒子边框线厚度为 \thinlines。 • 使用 \cornersize{x} 四个角的直径设为 x 乘以盒子宽和高之间的较小值。x 缺省为 0.5。 • 使用\cornersize*{x} 命令直接将四个角的直径设为 x。如 \cornersize*{1cm} 将四个角的直径设为 1 厘米。
\Ovalbox{Example} Example	除了盒子边框线厚度为 \thicklines 外，其它均与 \ovalbox 一样。

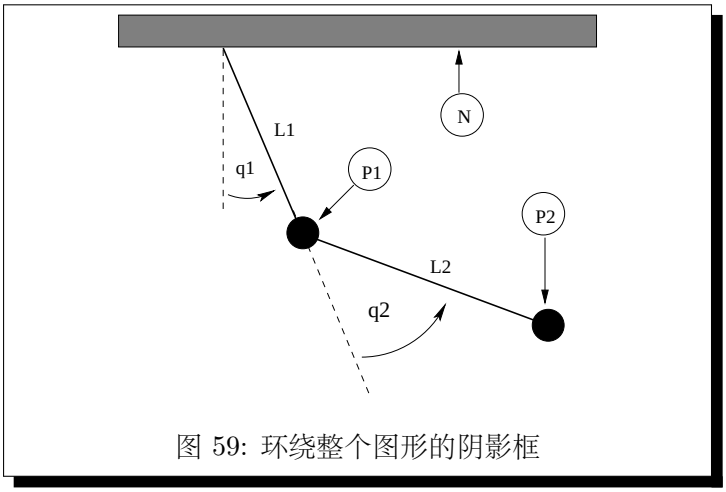


图 59: 环绕整个图形的阴影框

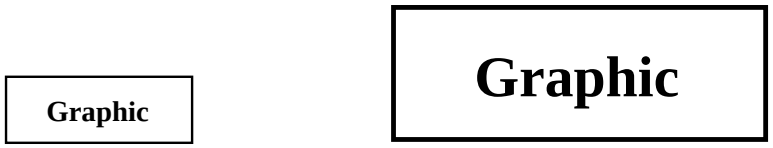


图 60: 一个 figure 环境中的两幅图像

第五部分　复杂图形

28　并列的图形

并列图形所需的命令取决于用户到底想怎样来组织图形。本节主要讨论三种常见的并列图形方式。

1. 多个图像并列于一个图形环境中。
2. 多个并列的浮动图形，如图 62 和 63。
3. 一图形环境中各个子图的平行排列。如子图 64a 和 64b 并列于图 64 中。

本节将用下列两种方法构建上述三种并列图形。

1. 连续使用 \includgraphics 命令。
2. 并列的小页环境，其中每个都包含一个 \includegraphics 命令。

在构造多个并列图形时，很重要的一点是要理解第 2 节的内容。并列图形是通过将若干盒子（\incudegraphics 或小页）平行放置在一条水平基线上来得到的。

28.1　单个图形环境中的并列图像

对于在单个 figure 环境中创建并列图像，尽管使用并列的小页环境能更容易地对齐图像，不过最简单的办法还是直接连续使用多个 \includgraphics 命令。

28.1.1　使用并列的 includegraphics 命令

如下代码：

```
\begin{figure}
    \centering
    \includegraphics[width=1in]{graphic}%
    \hspace{1in}%
    \includegraphics[width=2in]{graphic}
    \caption{一个 figure 环境中的两幅图像}
\end{figure}
```

会得到如图 60 的并列图形。该图形宽度为 4 英寸（第一幅图 1 英寸，\hspace 间距 1 英寸，第二幅图 2 英寸），居中放置。其中的 \hspace 命令可用 \hfill 来代替，这会将图形推向页面的两端边界（见第 10.2 节）。

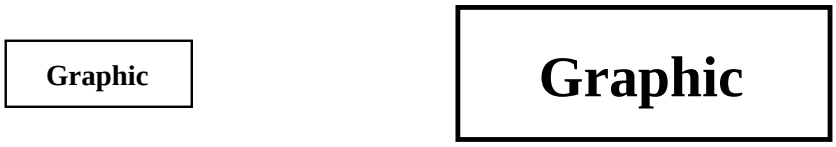


图 61: 中间对齐的图像

28.1.2 使用并列的小页环境

将 \includegraphics 命令放到小页环境中可以更好地控制图形的对齐方式。例如：

```
\begin{figure}
    \centering
    \begin{minipage}[c]{0.5\textwidth}
        \centering \includegraphics[width=1in]{graphic}
    \end{minipage}%
    \begin{minipage}[c]{0.5\textwidth}
        \centering \includegraphics[width=2in]{graphic}
    \end{minipage}
    \caption{中间对齐的图像}
\end{figure}
```

生成图 61 ，其中的图形是中间对齐的。

对于这个例子，需要注意以下几点：

- 如同其它的 LaTeX 对象一样，小页在放置时，它的参考点和当前基线对齐。缺省情况下小页使用 [c] 选项，将参考点置于其竖直方向的中点。选项 [t] 将参考点置于小页顶行的基线上，而选项 [b] 将参考点置于小页底行的基线上（参见第 11.4 节）。
- 在第一个 \end{minipage} 后面的 % 防止在两个小页之间多一个空格（参见第 10.2 节）。
- 当几个并列小页的宽度之和没有达到 1.0\textwidth 时，可用 \hspace 或 \hfill 来确定水平间距，详见第 10.2 节。

28.2 并列的浮动图形

在上一节中，在一个图形环境中使用多个小页环境可以得到一个由多幅图像组成的浮动图形。若将 \caption 命令放到每个小页环境中，则每个小页环境本身就变成浮动图形。例如：

```
\begin{figure}
    \centering
    %%----start of first figure----
    \begin{minipage}[t]{0.4\linewidth}
        \centering
        \includegraphics[width=1in]{graphic}
        \caption{Small Box} \label{fig:side:a}
    \end{minipage}%
    \hspace{1cm}%
    %%----start of second figure----
```

图 62: Small Box

图 63: Big Box

```
    \begin{minipage}[t]{0.4\linewidth}
        \centering
        \includegraphics[width=1.5in]{graphic}
        \caption{Big Box} \label{fig:side:b}
    \end{minipage}
\end{figure}
```

生成图 62 和 63。

关于该例子有几个注记：

- 尽管上面的命令只使用了一个 figure 环境，但由于每个小页中都包含 \caption 命令，所以仍然得到两个浮动图形。
- 两个放置图像的小页环境宽度为 figure 环境宽度的 40 %，之间有 1 厘米的水平间距。（注意，在 \end{minipage} 和 \hspace{1cm} 之后的注释会阻止额外的空格，从而确保了间距正好是 1 厘米。）默认情况下，图形标题的宽度就是小页环境的宽度。使用 1 厘米的水平间距是为了确保标题之间有空白（无论对长标题还是很宽的图像）。此外，标题的宽度也可以用 caption 宏包的 margin 或 width 关键字进行控制（参见表 18）。
- 紧接着 \begin{figure} 的 \centering 命令使得两个小页环境以及之间的空白在 figure 环境中居中放置。
- 小页内部的 \centering 命令使得图像在小页环境内部居中放置。

28.3 并列的子图形

在某些情况下，有时会希望将并列的图形组成一组，同时其中的每一幅图都保持其独立性。subcaption 宏包的 \subcaptionbox 命令（详见第 32 节）可以将一组独立的图像作为一个 figure 环境中的子图。例如：

```
\usepackage{subcaption}
...
\begin{figure}
    \centering
    %%----start of first subfigure----
    \subcaptionbox{Small Box with a Long Caption%
        \label{fig:subfig:a} %% label for first subfigure
        }{\includegraphics[width=1.1in]{graphic}}
    \hspace{1in}
    %%----start of second subfigure----
    \subcaptionbox{Big Box%
        \label{fig:subfig:b} %% label for second subfigure
        }{\includegraphics[width=1.5in]{graphic}}
```

(a) Small Box with a Long Caption

(b) Big Box

图 64: Two Subfigures

表 22: 图 64 的子图引用命令及其结果

引用命令	输出
\subref{fig:subfig:a}	a
\subref*{fig:subfig:a}	a
\ref{fig:subfig:a}	64a
\ref*{fig:subfig:a}	64a
\subref{fig:subfig:b}	b
\subref*{fig:subfig:b}	b
\ref{fig:subfig:b}	64b
\ref*{fig:subfig:b}	64b
\ref{fig:subfig}	64
\ref*{fig:subfig}	64

```
    \caption{Two Subfigures}
    \label{fig:subfig} %% label for entire figure
\end{figure}
```

生成图 64。表 22 展示了用于如何引用图 64 中子图的命令。其中带 * 的命令 \ref* 和 \subref* 需要导入 hyperref 宏包，效果是生成没有超链接的引用。

28.3.1 并列子图的宽度

使用 \subcaptionbox 创建的子图宽度默认为插入内容的自然宽度。由于子图 64a 只包含 \includegraphics 插图命令，因此其标题宽度取决于图片的宽度。\subcaptionbox 命令还提供了一个可选项用于控制子图的宽度，这样标题就可以和子图宽度一样了。例如，

```
\begin{figure}
    \centering
    %%----start of first subfigure----
    \subcaptionbox{Small Box with a Long Caption%
        \label{fig:specwid:subfig:a} %% label for first subfigure
    }[0.45\linewidth]{\includegraphics[width=1.1in]{graphic}}
    \hfill
    %%----start of second subfigure----
    \subcaptionbox{Big Box%
        \label{fig:specwid:subfig:b} %% label for second subfigure
    }[0.45\linewidth]{\includegraphics[width=1.5in]{graphic}}
    \caption{Two Subfigures}
```

(a) Small Box with a Long Caption

(b) Big Box

图 65: Two Subfigures

图 66: Box with a Long Caption

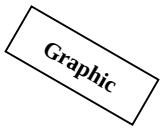


图 67: Rotated Box

```
    \label{fig:specwid:subfig} %% label for entire figure
\end{figure}
```

生成图 65，其中包含了子图 65a 和 65b。

29 标题中分开的小页环境

第 28.2 节描述了如何通过在小页环境中同时使用插图命令和 \caption 命令来构建并列图形。本节将介绍如何通过在不同的小页环境中使用插图命令和 \caption 命令来获得更好的对齐效果。

在图 62 和 63中，并列的小页环境使用了 [t] 选项，进而使得两幅图形的基线对齐。这对于非旋转的图形没有任何问题，而且使得两标题的顶部对齐。不过，如果图形的底部不对齐的话（如其中一图形被旋转），就会发生问题。例如：

```
\begin{figure}
    \centering
    %%----start of first figure----
    \begin{minipage}[t]{.4\linewidth}
        \centering
        \includegraphics[width=2cm]{graphic}
        \caption{Box with a Long Caption}
    \end{minipage}%
    \hspace{1cm}%
    %%----start of second figure----
    \begin{minipage}[t]{.4\linewidth}
        \centering
        \includegraphics[width=2cm,angle=-30]{graphic}
        \caption{Rotated Box}
    \end{minipage}%
\end{figure}
```

生成图 66 和 67，我们可以看到这里两幅图形的标题并不对齐。而若只使用小页的 [b] 选项，会使得标题的最后一行对齐，并不能解决问题。

一种解决办法是在创建两行小页环境，并把图形和标题分开放置：第一行放置图形，第二行放置标题。例如：

图 68: Box with a Long Caption

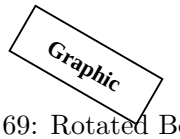


图 69: Rotated Box

```
\begin{figure}
    \centering
    %%----start of first figure graphics----
    \begin{minipage}[b]{.4\linewidth}
        \centering
        \includegraphics[width=2cm]{graphic}
    \end{minipage}%
    \hspace{1cm}%
    %%----start of second figure graphics----
    \begin{minipage}[b]{.4\linewidth}
        \centering
        \includegraphics[width=2cm,angle=-30]{graphic}
    \end{minipage}\\[-10pt]
    %%----start of first figure caption----
    \begin{minipage}[t]{.4\linewidth}
        \caption{Box with a Long Caption}
    \end{minipage}%
    \hspace{1cm}%
    %%----start of second figure caption----
    \begin{minipage}[t]{.4\linewidth}
        \caption{Rotated Box}
    \end{minipage}%
\end{figure}
```

生成的图 68 和 69 中，图形的基线和标题的第一行分别对齐。

在这个例子中，需要注意以下几点：

- 在最后一幅图后面用 \\ 来断行，\\ 的参数项 [-10pt] 使得图形与标题之间的距离比当前行距减少 10 pt。这样做是让图形和标题更接近些，用户也可自己选用合适的值。
- 包含图形的小页使用 [b] 选项，使得它们的参考点为其最后一行的基线。
- 包含标题小页使用 [t] 选项，使得它们的参考点为其第一行的基线，这样使得标题的顶行对齐。
- 任何一个 \label 命令都必须和它相应的 \caption 命令在同一个小页中。

30 图形与表格的平行排列

在第 28 节中，在一个 figure 环境中使用多个 \caption 命令可以得到并列的多个图形。同样地，在一个 table 环境中使用多个 \caption 命令可以得到并列的表格。

第 21 节介绍的 \captionof 命令可以将表格放在图形旁边。例如下面的命令：

```
\begin{figure}[htb]
    \begin{minipage}[b]{0.5\linewidth}
        \centering
        \includegraphics[width=0.8\linewidth]{graphic}
        \caption{This is a Figure by a Table}
        \label{fig:by:table}
    \end{minipage}%
    \begin{minipage}[b]{0.5\linewidth}
        \centering
        \begin{tabular}{|c|c|} \hline
            Day & Data \\ \hline\hline
            Monday & 14.6 \\
            Tuesday & 14.3 \\
            Wednesday & 14.2 \\
            Thursday & 14.5 \\
            Friday & 14.9 \\ \hline
        \end{tabular}
        \captionof{table}{This is a Table by a Figure}
        \label{table:by:fig}
    \end{minipage}
\end{figure}
```

使用一个 `figure` 环境创建图 70 和表 23。

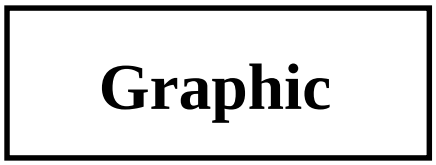


图 70: This is a Figure by a Table

Day	Data
Monday	14.6
Tuesday	14.3
Wednesday	14.2
Thursday	14.5
Friday	14.9

表 23: This is a Table by a Figure

因为 LaTeX 允许图形的浮动不必考虑其前后表格的顺序，所以在 `figure` 环境中使用

```
\captionof{table}{...}
```

可能会将该表格放在未处理的表格之前。类似地，在 `table` 环境中使用

```
\captionof{figure}{...}
```

可能会将该图形放在未处理的图形之前。如果不希望这样，可以在 `figure` 环境之前使用 `\FloatBarrier` 或者 `\clearpage` 命令清除未处理的浮动体（参见第 17.3 节）。

31 堆叠的图形和子图

第 28 节介绍了一系列方法创建并列图形。这些方法的原理都是将对象（图形、小页、子浮动体等）依次排列在一条直线上。当使用 `\\` 显式断行时，相同的方法也可以生成堆叠的图形。此外还可以通过 `\\` 命令的可选参数（例如 `\\[20pt]`）指定竖直间距。

31.1 堆叠的图形

第 28 节介绍了如何创建并列图形。本节的内容表明，添加断行可以生成多行图形。例如下列代码

```
\begin{figure}[htbp]
    \centering
    %%----start of first figure----
    \begin{minipage}[t]{0.25\linewidth}
        \centering
        \includegraphics[width=\linewidth]{graphic}
        \caption{First Stacked Figure}
        \label{fig:stacked:first}
    \end{minipage}%
    \hspace{1cm}%
    %%----start of second figure----
    \begin{minipage}[t]{0.25\linewidth}
        \centering
        \includegraphics[width=\linewidth]{graphic}
        \caption{Second Stacked Figure}
        \label{fig:stacked:second}
    \end{minipage}\\[20pt]
    %%----start of third figure----
    \begin{minipage}[t]{0.25\linewidth}
        \centering
        \includegraphics[width=\linewidth]{graphic}
        \caption{Third Stacked Figure}
        \label{fig:stacked:third}
    \end{minipage}%
    \hspace{1cm}%
    %%----start of fourth figure----
    \begin{minipage}[t]{0.25\linewidth}
        \centering
        \includegraphics[width=\linewidth]{graphic}
        \caption{Fourth Stacked Figure}
        \label{fig:stacked:fourth}
    \end{minipage}%
    \hspace{1cm}%
    %%----start of fifth figure----
    \begin{minipage}[t]{0.25\linewidth}
        \centering
        \includegraphics[width=\linewidth]{graphic}
        \caption{Fifth Stacked Figure}
        \label{fig:stacked:fifth}
    \end{minipage}%
\end{figure}
```

生成一组图 71–75。

图 71: First Stacked Figure

图 72: Second Stacked Figure

图 73: Third Stacked Figure

图 74: Fourth Stacked Figure

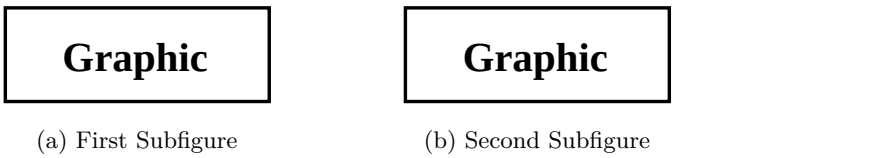
图 75: Fifth Stacked Figure

31.2 堆叠的子图

第 28.3 节介绍了如何创建并列的子图。本节的内容表明，添加断行可以生成多行子图。例如下列代码

```
\begin{figure}
    \centering
    %%----start of first subfigure----
    % label for first subfigure
    \subcaptionbox{First Subfigure\label{fig:stacksub:a}}%
        {\includegraphics[width=0.25\linewidth]{graphic}}
    \hspace{0.1\linewidth}
    %%----start of second subfigure----
    % label for second subfigure
    \subcaptionbox{Second Subfigure\label{fig:stacksub:b}}%
        {\includegraphics[width=0.25\linewidth]{graphic}}\\[20pt]
    %%----start of third subfigure----
    % label for third subfigure
    \subcaptionbox{Third Subfigure\label{fig:stacksub:c}}%
        {\includegraphics[width=0.25\linewidth]{graphic}}
    \hspace{0.1\linewidth}
    %%----start of fourth subfigure----
    % label for fourth subfigure
    \subcaptionbox{Fourth Subfigure\label{fig:stacksub:d}}%
        {\includegraphics[width=0.25\linewidth]{graphic}}
    \hspace{0.1\linewidth}
    %%----start of fifth subfigure----
    % label for fifth subfigure
    \subcaptionbox{Fifth Subfigure\label{fig:stacksub:e}}%
        {\includegraphics[width=0.25\linewidth]{graphic}}
    \caption{Five Subfigures}
    \label{fig:stacksub} %% label for entire figure
\end{figure}
```

生成图 76。



(a) First Subfigure (b) Second Subfigure

(c) Third Subfigure (d) Fourth Subfigure (e) Fifth Subfigure

图 76: Five Subfigures

32 Subcaption 宏包

第 28.3 节介绍了一个例子，其中使用 subcaption 宏包的 \subcaptionbox 命令创建子图，并且使用 \ref、\ref* 以及 subcaption 的 \subref、\subref* 对整个图形和每一子图进行标签引用[64]，相应的效果见表 22。

本节介绍 subcaption 宏包的更多知识，包括如何创建子浮动体以及如何进行标题定制。更多用法请参考宏包文档 [17]。

32.1 子浮动体的创建命令

subcaption 宏包依赖于 caption 宏包，使用时都要载入。subcaption 提供了三种方式可以创建子浮动体。

\subcaption 命令 在 \parbox 盒子或者 minipage 小页环境中使用 \subcaption 命令即可创建子浮动体。

```
\begin{figure}
...
\begin{minipage}{width}
    ...
    \subcaption[⟨list entry⟩]{⟨heading⟩}
\end{minipage}
...
\end{figure}
```

subfigure 和 subtable 环境 subcaption 宏包提供了 subfigure 和 subtable 两个环境用于快速创建子图和子表格，用法与 minipage 环境完全相同。在其中使用 \caption 命令可以直接创建子标题。例如

```
\begin{figure}
...
\begin{subfigure}[⟨pos⟩]{⟨width⟩}
    ...
    \caption{heading}
\end{subfigure}
```

[64] \ref* 由 hyperref 宏包提供，\subref* 同样需要 hyperref 宏包的支持。

```
...
\end{figure}
```

使用 subfigure 和 subtable 环境的好处是可以分别为子图和子表格设置不同的标题格式，同时也可以在环境的内部使用 \captionsetup 命令单独设置格式。

\subcaptionbox 命令 更为方便的是 \subcaptionbox 命令。语法为

\subcaptionbox[⟨目录标题⟩]{⟨标题⟩}[⟨宽度⟩][⟨盒子内位置⟩]{⟨内容⟩}

其中，⟨标题⟩ 中可以加入 \label 引用标签。⟨宽度⟩ 为可选项，默认为 ⟨内容⟩ 的自然宽度。⟨盒子内位置⟩ 表示 ⟨内容⟩ 在子浮动体内的对齐方式，默认为 c （居中 \centering），此外还可以是 r （居左 \raggedright）、l （居右 \raggedleft）和 s （无对齐格式）。

该命令创建的子浮动体按照首行的基线对齐，相当于 \parbox 盒子或者 minipage 环境的 [t] 选项。使用该命令的好处是无需指定宽度，即可按照自然宽度创建子浮动体，当然也可以手动指定宽度。相关例子见第 28.3 节。

32.2 子浮动体的标题定制

subcaption 宏包提供了 sub、subfigure 和 subtable 三种标题类型。子浮动体的标题设置包括以下几种方式，其中后面的设置方式会覆盖前面的方式：

- caption 宏包的设置。包括全局设置 \usepackage[⟨*options*⟩]{caption} 以及导言区和各 figure、table 浮动体环境中的 \captionsetup 命令设置等。
- 子浮动体标题类型 sub 的默认设置，具体为

  ```
  margin=0pt,font+=small,labelformat=parens,labelsep=space,
  skip=6pt,list=false,hypcap=false
  ```

  其中各个选项的含义参见第 20.2 节以及 caption 宏包文档 [16]。
- sub 类型的全局设置

  \usepackage[⟨*options*⟩]{subcaption}

  该命令等价于

  \usepackage{subcaption} \captionsetup[sub]{⟨*options*⟩}
- subfigure 和 subtable 的设置

  \captionsetup[subfigure]{⟨*options*⟩}
  \captionsetup[subtable]{⟨*options*⟩}

  这两个类型的设置适用于 subfigure 和 subtable 环境中的标题。
- 局部的 sub 类型设置，例如在 subfigure 和 subtable 环境内的设置。

  \captionsetup[sub]{⟨*options*⟩}

33 连续图形和连续子图

当连续两个图形的内容关系较为密切时，常常希望具有相同的图形编号。因为计数器 figure 中记录了下一图形的编号，所以可在图形环境前减小 figure 的值使得两幅图形具有相同的编号。例如：

```
\begin{figure}
....
\end{figure}
\addtocounter{figure}{-1}
\begin{figure}
....
\end{figure}
```

不过，这样做会使得两幅图形无法被正确区分，导致 LaTeX 的引用的混乱。

33.1 连续图形

连续图形既需要具有相同的编号，又应该有各自不同的引用，例如“Figure 12, Figure 12b”。构造连续图形的最佳办法是使用 caption 宏包。该宏包提供了 \ContinuedFloat 命令以及相应的标题类型 ContinuedFloat 以及同名的计数器 ContinuedFloat。

```
\renewcommand\theContinuedFloat{\alph{ContinuedFloat}}
\DeclareCaptionLabelFormat{continued}{Continued #1~#2}
\captionsetup[ContinuedFloat]{labelformat=continued}
...
\begin{figure}
    ...
    \caption{A figure}
\end{figure}
...
\begin{figure}\ContinuedFloat
    ...
    \caption{A figure}
\end{figure}
```

更多情况下希望第一幅图从“Figure 12a”而不是“Figure 12”开始，进而第二幅图是“Figure 12b”而不是“Figure 12a”。此时需要在第一幅图的环境内使用 \ContinuedFloat* 命令。

```
\renewcommand\theContinuedFloat{\alph{ContinuedFloat}}
...
\begin{figure}[tbp]
    \ContinuedFloat*
    \centering
    \includegraphics[height=6in]{tux-color}
    \caption{First figure of a series}
    \label{fig:continued:first}
\end{figure}
...
\begin{figure}[tbp]
```

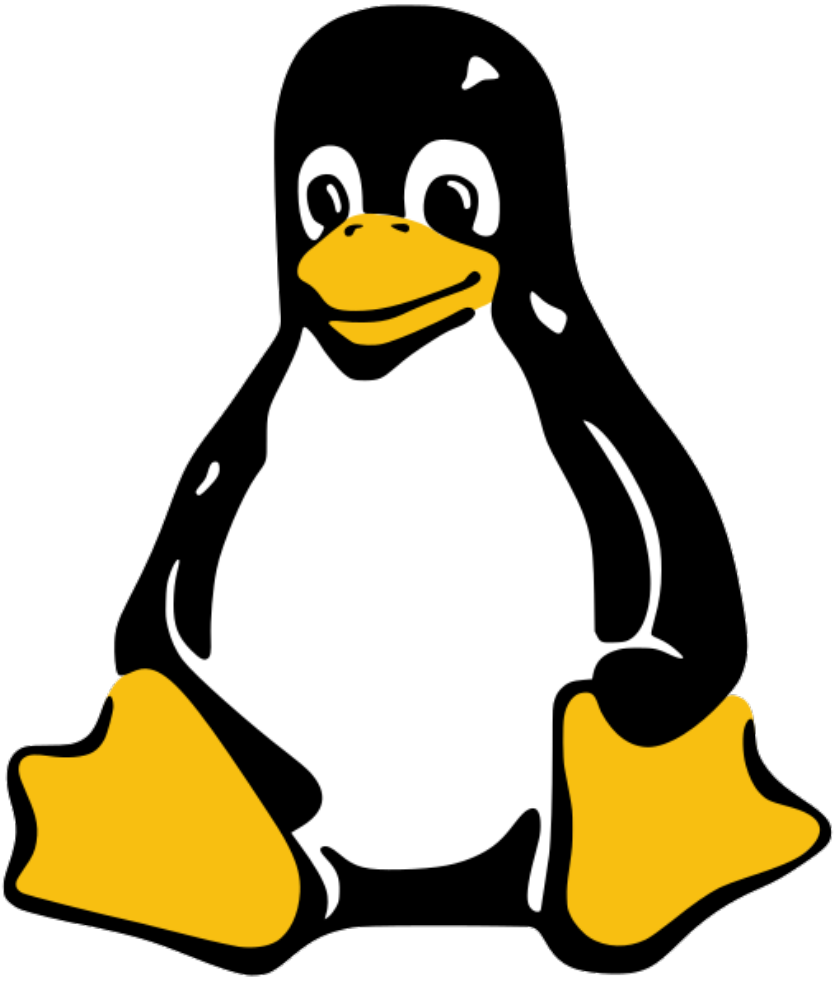


图 77a: First figure of a series

```
    \ContinuedFloat
    \centering
    \includegraphics[height=6in]{tux-black}
    \caption{Second figure of a series}
    \label{fig:continued:second}
\end{figure}
```

效果见图 77a 和图 77b。

33.2 连续子图

当一个 figure 环境中包含很多子图时，很多时候没有足够的空间将所有的子图放在一页内。此时使用 \ContinuedFloat 命令可以使子图分成若干组，并且具有相同的图形编号。这样就可以避免分割在不同编号的图形环境。

```
\usepackage{graphicx}
```

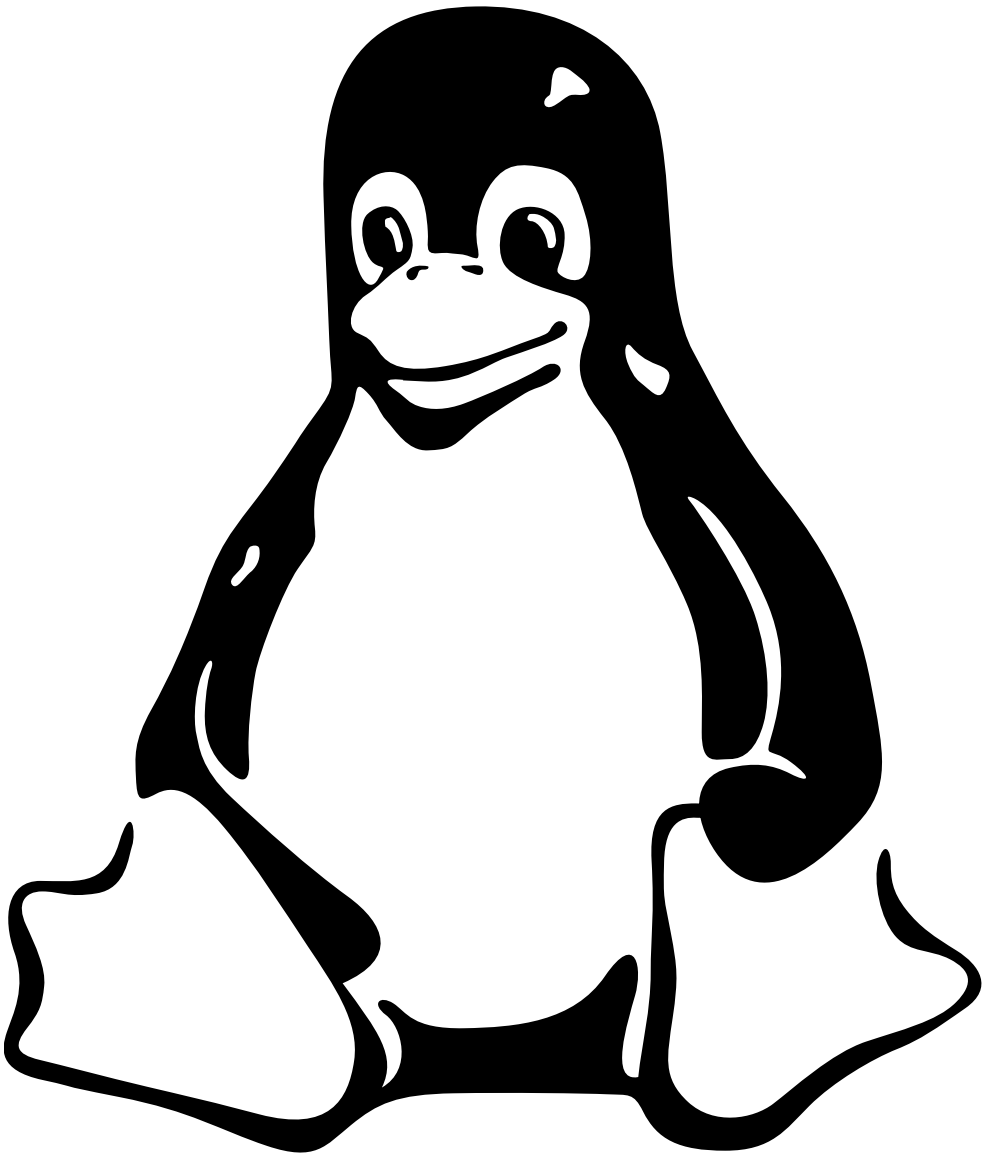


图 77b: Second figure of a series

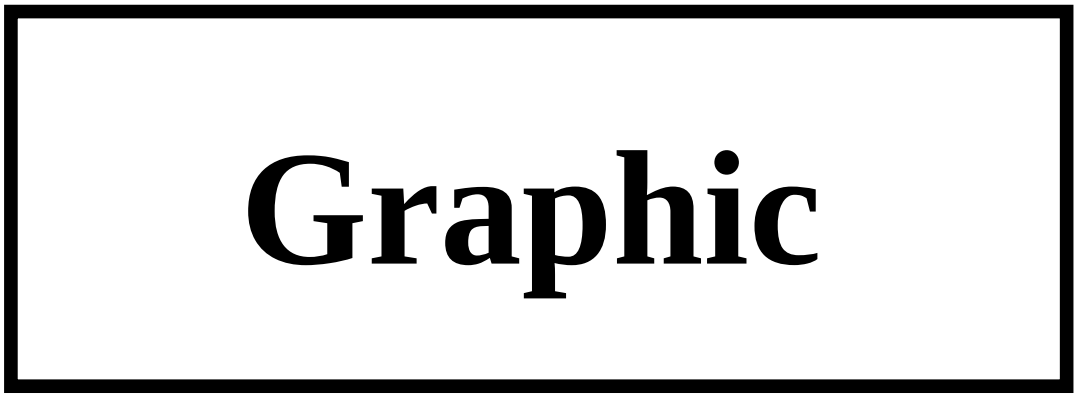
(a) First Subfigure (b) Second Subfigure

图 78: Two Subfigures

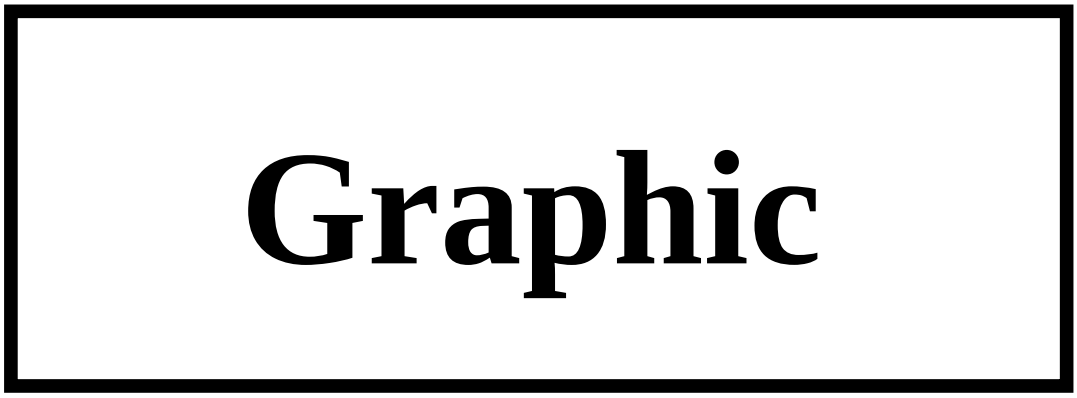
```
\usepackage{caption,subcaption}
...
\begin{figure}
    \centering
    %%----start of first subfigure----
    \subcaptionbox{%
        First Subfigure\label{fig:contfig:subone}}{%
        \includegraphics[width=3cm]{graphic}}
    \hspace{1cm}
    %%----start of second subfigure----
    \subcaptionbox{%
        Second Subfigure\label{fig:contfig:subtwo}}{%
        \includegraphics[width=3cm]{graphic}}
    \caption{Two Subfigures}
    \label{fig:contfig:one}
\end{figure}
\begin{figure}
    \ContinuedFloat
    \centering
    %%----start of third subfigure----
    \subcaptionbox{%
        Third Subfigure\label{fig:contfig:subthree}}{%
        \includegraphics[width=3cm]{graphic}}
    \hspace{1cm}
    %%----start of fourth subfigure----
    \subcaptionbox{%
        Fourth Subfigure\label{fig:contfig:subfour}}{%
        \includegraphics[width=3cm]{graphic}}
    \caption{Two Additional Subfigures}
    \label{fig:contfig:two}
\end{figure}
```

该代码片段会创建两个浮动体，分别包含图 78a、78b 以及图 78c、78d。显然这四个子图可以放在同一个浮动体内，但本例只是为了说明如何进行更大的子图分组。

注意 \ContinuedFloat 命令不仅保持浮动体编号不变，而且确保第二个浮动体内的子图编号不会重置为 (a)。此外两个浮动体的标题具有不同的引用标签（fig:contfig:one 和 fig:contfig:two），但相应的 \ref 引用值是相同的，不过由于两个浮动体可能不同页，因此对应的 \pageref 值可能不同。



(c) Third Subfigure



(d) Fourth Subfigure

图 78a: Two Additional Subfigures

34 图文混排

在使用外部图形时，通常的是将其置于一个 figure 环境中，由这一浮动环境来决定最后的位置是在页面的上方或下方。但有的时候，许多使用者往往希望将图形放置在一个正文方格内，或者置于页面的左右，也可能是在页面的的中间，四周包围者文本，甚至放在文字的下方作为背景，或重叠放置。这时，前面所介绍的只使用 LaTeX 图形宏包套件就很难得到所希望的结果。本节将介绍两个有用的图形宏包 wrapfig 和 picinpar，可以让你很容易地得到上述特殊效果。[65]

除了本节所介绍的宏包外，还有一些宏包也可完成同样的工作。如 floatflt[2] 以及较新的 cutwin[19] 也可用来将图形置于文本段落的一边。而所介绍的宏包中，也有未涉及的内容，进一步的研究可阅读这些宏包所附的帮助文件。

34.1 Wrapfig 宏包

Wrapfig 宏包提供了 wrapfigure 环境[66]来排版窄小的图形，使得该图形位于文本的一边，并使文本在其边上折行。

wrapfigure 的用法为：

```
\begin{wrapfigure}[⟨行数⟩]{⟨位置⟩}[⟨外延长度⟩]{⟨宽度⟩}
    ⟨图内容⟩
\end{wrapfigure}
```

这里 ⟨行数⟩ 是指图形高度所占的文本行的数目。如果不给出此选项，wrapfig 会自动计算。⟨位置⟩ 是指图形相对于文本的位置，须给定下面四项的一个。其中大写字母选项允许图形浮动，而小写字母选项则严格地将图形放在代码所在位置。

r,R 表示图形位于文本的右边。

l,L 表示图形位于文本的左边。

i,I 表示图形位于页面靠里的一边（用在 twoside 双面格式里）。

o,O 表示图形位于页面靠外的一边。

⟨外延长度⟩ 是指图形超出文本边界的长度，缺省为 0pt。如果大于 0pt，图形则会产生超出版心的效果。⟨宽度⟩ 则指图形的宽度。

在使用 wrapfig 环境进行图文绕排时需要注意下面几点：

- 图文绕排通常需要多次手动定位。由于会受到修改其它排版格式的影响，因此应当在最后定稿前进行调整。
- wrapfigure 环境不能放置在跨页的地方。
- 从美学角度讲，图文混排只应当用于普通文本。章节标题、大块的公式与图形混排会很难看。图形放在列

[65] 图文混排这一部分来自于旧译本，并经过一些修改调整。旧译本还介绍了 picins 宏包，但该宏包用于 LaTeX2.09，较新的发行版已无此宏包，故略去。——译注

[66] wrapfig 也同时提供了 wraptable 环境。

表的左边也很糟糕。（排版仍会允许，只是结果很差）如果是小型的公式则没有问题。

- 最好是在段落之间使用 wrapfigure 环境，如果想要在段落的内部使用，那么必须将 wrapfigure 环境放在可以自然断行的地方。
- 如果将 wrapfigure 放在 \parbox 或小页环境等分组中，文本折行必须在这些分组前结束。
- 在折行的文本中，\linewidth 并没有改变。

而浮动的绕排图形只会出现在以下之处：

- 段落开始的地方；
- 当前页面有足够的空间，或者可以在下一页继续；
- 文本不是标题或列表；
- 文本没有环绕其它图形；
- 正文中的文本，而不是小页环境。

本节中的例子使用了如下命令：

```
\begin{wrapfigure}{r}{4.5cm}
\includegraphics[width=4cm,clip]{tiger}
\end{wrapfigure}
在使用 \env{wrapfig} 环境进行图文绕排时需要注意下面几点：...
```

34.2 Picinpar 宏包

picinpar 宏包定义了一个基本的环境 window，以及两个变体环境 figwindow 和 tabwindow。效果是在文本段落中打开一个“窗口”，在其中放入图形、文字和表格等。这里我们主要讨论将图形放入文本段落的用法，其它的用法可参考 picinpar 的说明。

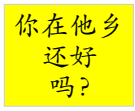
```
\begin{window}[⟨下降行数⟩,⟨水平位置⟩,⟨内容⟩,⟨内容说明⟩]
    ⟨绕排文字⟩
\end{window}

\begin{figwindow}[⟨下降行数⟩,⟨水平位置⟩,⟨图形⟩,⟨图标题⟩]
    ⟨绕排文字⟩
\end{figwindow}
```

这里的 ⟨下降行数⟩ 是指“窗口”开始前的行数。⟨水平位置⟩ 是指在段落中“窗口”的位置，缺省为 l，即居左。另外两种是 c（居中）和 r（居右）。第三个参数 ⟨内容⟩ 是出现在“窗口”中的内容，这在 figwindow 中就是要插入的图形。第四个参数 ⟨内容说明⟩ 则是对“窗口”内容的说明性文字，这在 figwindow 中就是图形的标题。下面是几个例子：

```
\begin{window}[2,c,{\fcolorbox{morelight}{yellow}{%
    \shortstack{你在他乡 \\还 好 \\ 吗？}}},{}]
    可是哈卜拉姆再聪明...可是我偏不喜欢。」
\end{window}
```

可是哈卜拉姆再聪明、再有学问，有一件事却是他不能解答的，因为包罗万象的「可兰经」上也没有答案；如果你深深爱著的人，却深深的爱上了别人，有甚麼法子？白马带著她一步步的回到中原。白马已经老了，只能慢慢的走，但终是能回到中原的。江南有杨柳、桃花，有燕子、金鱼……汉人中有的是英俊勇武的少年，倜傥潇洒的少年……但这个美丽的姑娘就像古高昌国人那样固执：「那都是很好很好的，可是我偏不喜欢。」




```
\begin{figwindow}[1,r,{\mbox{%
    \includegraphics[width=2cm]{tiger}}},{Tiger}]
    可是哈卜拉姆再聪明...可是我偏不喜欢。」
\end{figwindow}
```

可是哈卜拉姆再聪明、再有学问，有一件事却是他不能解答的，因为包罗万象的「可兰经」上也没有答案；如果你深深爱著的人，却深深的爱上了别人，有甚麼法子？白马带著她一步步的回到中原。白马已经老了，只能慢慢的走，但终是能回到中原的。江南有杨柳、桃花，有燕子、金鱼……汉人中有的是英俊勇武的少年，倜傥潇洒的少年……但这个美丽的姑娘就像古高昌国人那样固执：「那都是很好很好的，可是我偏不喜欢。」

图 79: Tiger

```
\begin{figwindow}[1,c,{\mbox{%
    \includegraphics[width=2cm]{tiger}}},{Tiger}]
    可是哈卜拉姆再聪明...可是我偏不喜欢。」
\end{figwindow}
```

可是哈卜拉姆再聪明、再有学问，有一件事却是他不能解答的，因为包罗万象的「可兰经」上也没有答案；如果你深深爱著的人，却深深的爱上了别人，有甚麼法子？白马带著她一步步的回到中原。白马已经老了，只能慢慢的走，但终是能回到中原的。江南有杨柳、桃花，有燕子、金鱼……汉人中有的是英俊勇武的少年，倜傥潇洒的少年……但这个美丽的姑娘就像古高昌国人那样固执：「那都是很好很好的，可是我偏不喜欢。」

图 80: Tiger

在使用 picinpar 时要注意以下几点：

- 不要在 window 环境中使用 \samepage。
- 不要在 window 环境中使用 \footnote，代之用 \footnotemark 标记角注，而将角注的内容在 window 环境外用 \footnotetext 加入。

参考文献

[1] David Carlisle. *Packages in the 'graphics' bundle*. 2016. URL: http://mirrors.ctan.org/macros/latex/required/graphics/grfguide.pdf.

[2] Mats Dahlgreny. *Welcome to the floatflt package!* 1998. URL: http://mirrors.ctan.org/macros/latex/contrib/floatflt/floatflt.pdf.

[3] Robin Fairbairns. *The topcapt package*. 2004. URL: http://www.ctan.org/tex-archive/macros/latex/contrib/misc/topcapt.sty.

[4] Michel Goossens, Sebastian Rahtz, and Frank Mittelbach. *The LaTeX Graphics Companion*. Reading, Massachusetts: Addison-Wesley, 1997. ISBN: 0-201-85469-4.

[5] Michael C. Grant and David Carlisle. *The PSfrag package, version 3*. 1998. URL: http://mirrors.ctan.org/macros/latex/contrib/psfrag/pfgguide.pdf.

[6] John D. Hobby. *METAPOST A User's Manual*. 2014. URL: http://mirrors.ctan.org/systems/doc/metapost/mpman.pdf.

[7] Leslie Lamport. *LaTeX: A Document Preparation System*. 2nd ed. Reading, Massachusetts: Addison-Wesley, 1994. ISBN: 0-201-52983-1.

[8] Gonzalo Medina. *The background package*. 2014. URL: http://mirrors.ctan.org/macros/latex/contrib/background/background.pdf.

[9] Frank Mittelbach. *The varioref package*. 2016. URL: http://mirrors.ctan.org/macros/latex/required/tools/varioref.pdf.

[10] Frank Mittelbach et al. *The LaTeX Companion*. 2nd ed. Boston, Massachusetts: Addison-Wesley Pearson Education, 2004. ISBN: 0-201-36299-6.

[11] Ahmed Musa. *The xwatermark package*. 2012. URL: http://mirrors.ctan.org/macros/latex/contrib/xwatermark/xwatermark-doc.pdf.

[12] Rolf Niepraschk. *The eso-pic package*. 2015. URL: http://mirrors.ctan.org/macros/latex/contrib/eso-pic/eso-pic.pdf.

[13] Piet van Oostrum. *Page layout in LaTeX*. 2016. URL: http://mirrors.ctan.org/macros/latex/contrib/fancybox/fancybox-doc.pdf.

[14] Sebastian Rahtz and Heiko Oberdiek. *Hypertext marks in LaTeX: a manual for hyperref*. 2012. URL: http://mirrors.ctan.org/macros/latex/contrib/hyperref/doc/manual.pdf.

[15] Will Robertson. *The pstool package*. 2014. URL: http://mirrors.ctan.org/macros/latex/contrib/pstool/pstool.pdf.

[16] Axel Sommerfeldt. *Customizing captions of floating environments*. 2011. URL: http://mirrors.ctan.org/macros/latex/contrib/caption/caption-eng.pdf.

[17] Axel Sommerfeldt. *The subcaption package*. 2013. URL: http://mirrors.ctan.org/macros/latex/contrib/caption/subcaption.pdf.

[18] Peter Wilson and Herries Press. *The ccaption package*. 2011. URL: http://mirrors.ctan.org/macros/latex/contrib/ccaption/ccaption.pdf.

[19] Peter Wilsony and Alan Hoenig. *Making cutouts in paragraphs*. 2010. URL: http://mirrors.ctan.org/macros/latex/contrib/cutwin/cutwin.pdf.

索引