

# L<sup>A</sup>T<sub>E</sub>X 排版学习笔记

zoho@bbs.ctex.org

2016 年 10 月 24 日

# 前言

从 2006 年 3 月 29 日开始，陆陆续续记录了排版 LaTeX 文档时的笔记。直到 2011 年 11 月 6 日大概有了 30 多篇记录，于是开始将所有内容整理为一个 PDF 文档。然后继续增添内容，到目前为止，总共有 100 页。

# 目录

第一章 基本使用	7
1.1 基本排版流程	7
1.2 文本排版	7
1.2.1 英文文档	7
1.2.2 特殊字符	8
1.2.3 段落换行	9
1.2.4 中文文档	9
1.3 文本环境	10
1.3.1 列表环境	10
1.3.2 对齐环境	10
1.3.3 定理环境	12
1.4 文档结构	13
1.4.1 文档类别	13
1.4.2 标题摘要	14
1.4.3 章节目录	14
1.4.4 参考文献	15
1.4.5 全文索引	16
1.5 文本盒子	16
1.5.1 左右盒子	16
1.5.2 段落盒子	18
1.5.3 标尺盒子	20
1.5.4 幻影盒子	21
1.5.5 盒子变换	21
1.6 插入表格	23

1.6.1	基本表格	23
1.6.2	跨列表格	24
1.6.3	浮动表格	24
1.7	插入图片	25
1.7.1	图文并排	25
1.7.2	图文分开	26
1.7.3	浮动图片	27
<b>第二章</b>	<b>格式调整</b>	<b>29</b>
2.1	空白调整	29
2.1.1	长度单位	29
2.1.2	行距调整	29
2.1.3	弹性长度	30
2.1.4	表格行距	31
2.2	字体使用	31
2.2.1	字体编码	31
2.2.2	字体族名	31
2.2.3	字体系列	32
2.2.4	字体形状	32
2.2.5	字体大小	33
2.3	页面大小布局	33
2.4	目录书签	34
2.4.1	添加目录项	34
2.4.2	添加书签项	35
2.4.3	替换书签项	36
2.4.4	目录页格式	36
<b>第三章</b>	<b>数学公式</b>	<b>38</b>
3.1	两种公式	38
3.2	各种字母	39
3.3	数学函数	40
3.4	配对括号	40
3.5	多行公式	41
3.6	画交换图	43

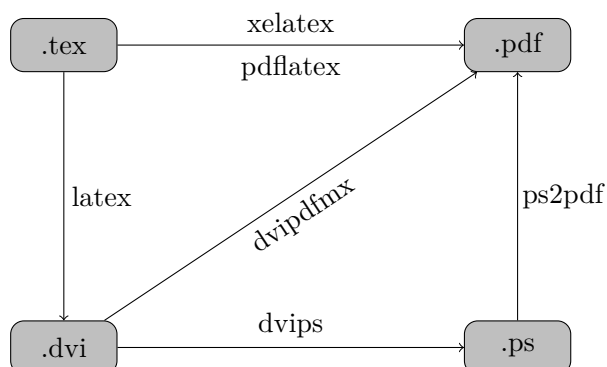
目录	5
3.6.1 用 amscd 包画交换图	43
3.6.2 用 xy-pic 包画交换图	43
3.7 公式行距	44
<b>第四章 图形绘制</b>	<b>46</b>
4.1 画图语言和画图软件	46
4.2 PGF/TikZ 绘图	46
4.2.1 画流程图	47
4.2.2 函数图像	47
4.2.3 几何图形	48
<b>第五章 演示文稿</b>	<b>50</b>
5.1 最简单例子	51
5.2 逐步显示	52
5.3 主题模板	53
5.4 中文演示	54
<b>第六章 高级应用</b>	<b>56</b>
6.1 定义命令和环境	56
6.2 使用计数器	57
6.3 处理错误和警告	58
6.3.1 编译时的错误信息	59
6.3.2 编译时的警告信息	59
<b>第七章 字体详解</b>	<b>61</b>
7.1 字体类型与文档格式	61
7.2 新字体选择方案 NFSS	62
7.3 中文排版历程	63
7.3.1 CCT 中文排版	63
7.3.2 CJK 中文排版	64
7.3.3 XeTeX 中文排版	65
<b>附录 A 常用宏包介绍</b>	<b>67</b>

<b>附录 B TeX 背景资料</b>	<b>76</b>
B.1 TeX 系统的前世今生 . . . . .	76
B.2 TeX 系统的目录结构 . . . . .	77
B.3 TeX 系统的文件查找 . . . . .	78
<b>附录 C TeX 排版原理</b>	<b>81</b>
C.1 读取文件 . . . . .	81
C.2 生成盒子 . . . . .	82
C.3 分段为行 . . . . .	83
C.4 组行为页 . . . . .	85
<b>附录 D TeX 相关软件</b>	<b>87</b>
D.1 纯文本编辑器 . . . . .	87
D.1.1 TeXMaker 编辑器 . . . . .	87
D.1.2 TeXworks 编辑器 . . . . .	88
D.1.3 正向搜索与反向搜索 . . . . .	89
D.2 可视化编辑器 . . . . .	90
D.2.1 LyX 文档处理软件 . . . . .	90
D.2.2 TeXmacs 文档排版软件 . . . . .	92
D.3 新版 Office 的公式排版 . . . . .	93
D.4 在网页中显示数学公式 . . . . .	94
<b>参考文献</b>	<b>96</b>
<b>全书索引</b>	<b>97</b>

# 第一章 基本使用

## 1.1 基本排版流程

和 Microsoft Office Word 等所见即所得的办公软件不同，用  $\text{\LaTeX}$  排版文档，首先要用文本编辑器编辑好 `tex` 文档，然后通过各种程序编译，得到 `pdf` 文档用于打印或者阅读。基本的排版流程如下图：



一般我们经常用 `pdflatex` 或者 `xelatex` 程序直接从 `tex` 文件生成 `pdf` 文件。如果是中文 `tex` 文档，**优先使用 `xelatex` 程序编译**。如何使用编辑器以及编译文档在后面的章节有详细介绍。

## 1.2 文本排版

### 1.2.1 英文文档

最简单的  $\text{\LaTeX}$  英文文档如下：

```
% hello.tex
\documentclass[a4paper]{article}
\usepackage{hyperref}
\begin{document}
Hello World!
\end{document}
```

我们来看看上面的 tex 文档内容。在 L<sup>A</sup>T<sub>E</sub>X 里面，每行的 % 符号后面的内容都表示注释，在编译时都会被忽略。因此第一行的内容没有任何结果。

接着看第二行。在 L<sup>A</sup>T<sub>E</sub>X 文档中，用 \ 开始的字母串来表示一个命令，这里的 \documentclass 是 L<sup>A</sup>T<sub>E</sub>X 文档的基本命令，用于指明文档类。命令后面用花括号 {} 包含的内容是该命令的参数，必不可少。对于这个例子，article 参数就指明我们撰写的是一篇文章，类似地还可以用 book 或者 report 参数，分别表示书籍和报告。命令后用方括号 [] 包含的内容是该命令的选项，选项可以省略，省略时必需连同方括号也去掉。如果选项省略，将会使用默认值。这里例子的 a4paper 指明我们文档的页面是 A4 纸张的大小，改为 b5paper 就是 B5 纸张的大小。

第三行的 \usepackage 命令也是 L<sup>A</sup>T<sub>E</sub>X 的基本命令，用于载入 L<sup>A</sup>T<sub>E</sub>X 宏包。L<sup>A</sup>T<sub>E</sub>X 系统中包含了各种各样的宏包，对 L<sup>A</sup>T<sub>E</sub>X 的基本功能作了各种扩展。我们这一行载入的 hyperref 宏包后面将介绍到。

接下来在 \begin{document} 和 \end{document} 之间的部分我们称为正文区，一般用于正文内容的撰写。这个例子的正文内容只有简单的 Hello World!。对应的，在 \documentclass 和 \begin{document} 命令之间的部分我们称为导言区，一般用于载入宏包，定义命令和调整格式。L<sup>A</sup>T<sub>E</sub>X 文档把格式和内容部分分开，是一种良好的设计准则。

1.2.2 特殊字符

在 L<sup>A</sup>T<sub>E</sub>X 的文本内容中，大部分字符都可以直接输入，但是 #, \$, %, &, {, }, \_, ^, ~, <, >, |, \ 这几个字符由于有特殊用途不能直接输入。我们可以按照下表的方式输入这些特殊字符

输入	\#	\\$	\%	\&	\{	\}	\_	\^{}	\~{}
显示	#	\$	%	&	{	}	_	^	~
输入	\textless		\textgreater		\textbar		\textbackslash		
显示	<		>				\		



另外，LaTeX 中是区分左右引号的，输入 ``` 和 `‘` 分别显示左右单引号，输入 ```` 和 `“` 分别显示左右双引号。连字号、起止号和破折号看起来都是一个差不多的短横线，但输入的方式也是不同的。用 `-` 得到连接单词的连字号-，用 `--` 得到表示数字范围的起止号—，用 `---` 得到英文的破折号—。

### 1.2.3 段落换行

用一个空行或者 `\par` 命令可以开始新的段落，同时会有默认的首行缩进。用 `\\` 或者 `\newline` 可以强制换行在下一行继续，且在下一行不会有缩进。

### 1.2.4 中文文档

LaTeX 中文文档的排版有各种方式，例如 CCT, CJK, xeCJK 等等。目前最优秀的方式是用 `ctex` 文档类来排版中文文档，它在其它各种方式的基础上以一致的方式解决了中文排版的问题。例如：

```
\documentclass[UTF8]{ctexart}
\begin{document}
中文内容测试！
\end{document}
```

使用这种方式，只需要将文档类从英文的 `article` 改成 `ctexart`，所有中文环境和章节编号等等都已经按照中文习惯设置好了，简单易行。

例子中的 `UTF8` 这个可选参数指明了中文文档的编码。编码主要有这两种：`GBK` 和 `UTF8`；如果编码不指定，默认为 `GBK`。不同的 LaTeX 编辑器对中文文档的默认编码不同。以 CTeX 套装 2.9 版本包含的两个编辑器为例，WinEdt 编辑器的默认中文编码为 `GBK`，而 TeXworks 编辑器的默认中文编码为 `UTF8`。

因此，对于初学者，为了避免乱码的出现，最好根据编辑器的默认情形来选择编码，如果使用的是 WinEdt 编辑器，将文档的编码设为 `GBK`，如果使用的是 TeXworks 编辑器，将文档的编码设为 `UTF8`。

如果希望在不同的编辑器都能正常显示中文，推荐统一使用 `UTF8` 编码。在 WinEdt 中新建一个空白文件，在第一行写上如下内容

```
% -*- coding: utf-8 -*-
```

然后保存并关闭，再重新打开后，将旧文档内容复制粘贴到里面，保存以后该文档就保持为 `UTF8` 编码了。

另外，对于 `book` 和 `report` 文档类，也有对应的 `ctexbook` 和 `ctexrep` 中文文档类，其用法类似。

## 1.3 文本环境

在  $\text{\LaTeX}$  中，用 `\begin` 和 `\end` 包含起来的特殊段落，即

```
\begin{环境名}...\end{环境名}
```

我们称为环境。不同环境中的内容将用不同的方式来排版。在文本段落中，常见的环境有列表环境，对齐环境和定理环境等。

### 1.3.1 列表环境

列表环境有三种：无序列表（`itemize`）、有序列表（`enumerate`）和描述列表（`description`）。它们的使用方式和显示效果如下：

```
\begin{itemize}
\item 无编号的列表
\item 带编号的列表
\item 带标签的列表
\end{itemize}
```

- 无编号的列表
- 带编号的列表
- 带标签的列表

```
\begin{enumerate}
\item 无编号的列表
\item 带编号的列表
\item 带标签的列表
\end{enumerate}
```

1. 无编号的列表
2. 带编号的列表
3. 带标签的列表

```
\begin{description}
\item[无序列表] 无编号的列表。
\item[有序列表] 带编号的列表。
\item[描述列表] 带标签的列表。
\end{description}
```

**无序列表** 无编号的列表。  
**有序列表** 带编号的列表。  
**描述列表** 带标签的列表。

### 1.3.2 对齐环境

默认情形，文章段落都是两侧对齐的，但我们也可以排版居中对齐和单侧对齐的段落。

### 1.3.2.1 居中对齐

在 LaTeX 中, 可以用 `center` 环境得到居中的文本段落, 其中可以用 `\\` 换行。例如:

```
This is the first normal paragraph. This is the first normal paragraph.  
\begin{center}  
Hi, the center. Hi, the center. Hi, the center. \\  
Hi, the center. Hi, the center. Hi, the center.  
\end{center}  
This is another normal paragraph. This is another normal paragraph.
```

This is the first normal paragraph. This is the first normal paragraph.

Hi, the center. Hi, the center. Hi, the center.  
Hi, the center. Hi, the center. Hi, the center.

This is another normal paragraph. This is another normal paragraph.

如果居中段落在一行放不下, 只会在最后一行是居中的, 其它行都填满页面宽度。在一个环境内部, 用命令 `\centering` 也可以让后面的文本都居中放置。

### 1.3.2.2 单侧对齐

类似地, 可以用 `flushleft` 和 `flushright` 环境分别得到向左对齐和向右对齐的文本段落。例如:

```
This is the first normal paragraph. It is justified.  
\begin{flushleft}  
Hi, the left. Hi, the left. Hi, the left. Hi, the left. \\  
Hi, the left. Hi, the left. Hi, the left. Hi, the left.  
\end{flushleft}  
\begin{flushright}  
Hi, the right. Hi, the right. Hi, the right. Hi, the right. \\  
Hi, the right. Hi, the right. Hi, the right. Hi, the right.  
\end{flushright}  
This is the first normal paragraph. It is justified.
```

This is the first normal paragraph. It is justified.

Hi, the left. Hi, the left. Hi, the left. Hi, the left.

Hi, the left. Hi, the left. Hi, the left. Hi, the left.

Hi, the right. Hi, the right. Hi, the right. Hi, the right.

Hi, the right. Hi, the right. Hi, the right. Hi, the right.

This is the first normal paragraph. It is justified.

类似地，在一个环境内部，用 `\raggedright` 声明同样可以让文本向左对齐，而用 `\raggedleft` 声明同样可以让文本向右对齐。<sup>1</sup>

### 1.3.3 定理环境

定理命题的撰写的最简单的例子：

```
\newtheorem{theorem}{Theorem}
\newtheorem{corollary}{Corollary}
\begin{theorem}
This is a theorem.
\end{theorem}
\begin{corollary}
This is a corollary.
\end{corollary}
```

**Theorem 1** *This is a theorem.*

**Corollary 1** *This is a corollary.*

这样的输出结果就是各自编号的定理和推论了，定义、命题等等也类似可以这么使用。如果你希望定理和推论一起编号，前面可以改为这样（意思是 corollary 也使用 theorem 的编号）：

```
\newtheorem{thrm}{Theorem}
\newtheorem{corl}[thrm]{Corollary}
\begin{thrm}
This is a theorem.
\end{thrm}
\begin{corl}
This is a corollary.
\end{corl}
```

**Theorem 1** *This is a theorem.*

**Corollary 2** *This is a corollary.*

<sup>1</sup>在英文中，ragged right 的字面意思是“右侧不对齐”。

最后, 如果你希望使用类似 Theorem 3.6.1 这样的定理编号, 可以这么使用:

```
\newtheorem{thm}{Theorem}[section]
\newtheorem{cor}[thm]{Corollary}
\begin{thm}
This is a theorem.
\end{thm}
\begin{cor}
This is a corollary.
\end{cor}
```

**Theorem 1.3.1** *This is a theorem.*

**Corollary 1.3.2** *This is a corollary.*

这里的例子是对于 book 文档类来说的, 如果是 article 文档类, 最前面的 section 应改为 subsection, 否则将输出类似 Theorem 6.1 这样的编号。

## 1.4 文档结构

### 1.4.1 文档类别

前面已经讲过, 用下面的命令可以指定文档类

```
\documentclass[选项]{文档类}
```

其中文档类有文章 (article), 书籍 (book), 报告 (report), 书信 (letter) 这几种选择。文档类的选项有下面这些。

页面大小 (a4paper, a5paper, b4paper, letterpaper, legalpaper, executivepaper): 默认的 letterpaper 常见于美国, 其大小为 216 毫米乘以 279 毫米 (即 8.5 英寸乘以 11 英寸), 而国内常用的 a4paper 的大小为 210 毫米乘以 297 毫米, 两者稍有区别。虽然这里可以指定不同的页面大小, 但是纸张却始终为 A4 大小。比如你指定为 A5 页面大小, 那么生成的 PDF 文件中, 内容只在 A4 纸张的左上角。这个页面大小和纸张大小不一致的问题非常烦人, 因此建议不在这里设置页面大小, 而是用 geometry 宏包来设置 (见第 2.3 节)。

字体大小 (10pt, 11pt, 12pt): 默认为 10pt。

纸张方向 (portrait, landscape): 默认为 portrait (纵向), 在屏幕阅读也许 landscape (横向) 更方便。

草稿定稿 (draft, final): 默认为 final (定稿); 如果是 draft (草稿), 页面内容有溢出时会显示粗黑条。

单面双面 (oneside, twoside): 对于 article 和 report 文档类, 默认设置为单面, 页码总是在右边; 对于 book 文档类, 默认设置为双面, 奇数页页码在右边, 偶数页页码在左边, 这样双面打印时页码总在外侧。

新章开始 (openright, openany): 仅对 book 文档类有效, 默认值为 openright, 即每章都从奇数页开始; 如果设置为 openany, 则每章仅从新的一页开始, 不管奇偶页。

### 1.4.2 标题摘要


用下面的代码可以加入文章的标题、作者和日期信息:

```
\title{Notes On LaTeX Typesetting}
\author{Some One}
\date{November 13, 2011}
\maketitle
```

如果 \date{} 命令的参数为空, 则不显示日期信息。如果不出现 \date 命令, 则默认显示当前的日期。

用下面的代码可以加入文章的内容摘要:

```
\begin{abstract} only 'article' has 'abstract'. 'Book' and 'Report' don't!
some abstract...
\end{abstract}
```



### 1.4.3 章节目录

在 book 和 report 文档类中, 可以使用 \part、\chapter、\section、\subsection、\subsubsection、\paragraph、\subparagraph 这些章节命令, 在 article 文档类中, 除了 \chapter 不能用, 其它的都可以用。例如下面的代码

```
\part{部分标题}
\chapter{章标题}这一章我们介绍这些内容。
\section{节标题}这一节我们介绍这些内容。
\subsection{小节标题}这一小节我们介绍这些内容。
\subsubsection{子节标题}这一子节我们介绍这些内容。
\paragraph{段标题}这一段我们介绍这些内容。
\subparagraph{小段标题}这一小段我们介绍这些内容。
```

用 `\tableofcontents` 命令可以自动从各章节标题生成目录，但此时需要编译两次。<sup>2</sup> 上面各个章节命令都对应有带 \* 号的命令（例如 `\section*`，`\subsection*` 等等），这些带 \* 号的章节标题将不会自动编号，也不会出现在目录中。

在导言区中用下面的命令载入 `hyperref` 宏包

```
\usepackage{hyperref}
```

就可以让生成的文章目录有链接，点击时会自动跳转到该章节。而且也会使得生成的 pdf 文件带有目录书签。

#### 1.4.4 参考文献

在 LaTeX 中使用参考文献很容易，下面的代码

```
\begin{thebibliography}{123456}
\bibitem[Knuth1]{DK1} D. Knuth, T.A.O.C.P. , Vol. 1, Addison-Wesley, 1997.
\bibitem[Knuth2]{DK2} D. Knuth, T.A.O.C.P. , Vol. 2, Addison-Wesley, 1997.
\bibitem[Knuth3]{DK3} D. Knuth, T.A.O.C.P. , Vol. 3, Addison-Wesley, 1998.
\end{thebibliography}
```

将得到下面的结果：

### 参考文献

[Knuth1] D. Knuth, T.A.O.C.P. , Vol. 1, Addison-Wesley, 1997.  
 [Knuth2] D. Knuth, T.A.O.C.P. , Vol. 2, Addison-Wesley, 1997.  
 [Knuth3] D. Knuth, T.A.O.C.P. , Vol. 3, Addison-Wesley, 1998.

其中方括号里的可选参数是结果中显示的各条文献记号，如果省略则使用数字编号。它后面大括号里的必选参数是引用的名称，要引用前两条文献，在前面可以这样使用：

```
D. Knuth wrote some books, e.g. \cite{DK1, DK2}.
```

在 `\begin{thebibliography}` 后面的必选参数指明各文献记号的最大长度，即它的字符长度应该等于各文献记号的最大长度。

<sup>2</sup>类似地，命令 `\listoffigures` 和 `\listoftables` 分别生成插图目录和表格目录。

1.4.5 全文索引

利用 `makeindex` 宏包可以建立全文索引。最简单的例子如下：

```
\documentclass{article}
\usepackage{makeidx}
\makeindex
\begin{document}
...
Plain TeX \index{Plain TeX}
...
LaTeX \index{LaTeX}
...
ConTeXt \index{ConTeXt}
...
\printindex
\end{document}
```

generate a my.idx, makeindex.exe my.idx

在编译时，先用 xelatex/pdflatex 编译，接着用 `makeindex` 编译，最后再用 `xelatex/pdflatex` 编译，才能得到带全文索引的文档。

1.5 文本盒子

在 `TeX` 排版文档时，所有内容都会被转换为嵌套的盒子。每个字符是一个盒子，多个字符盒子组成一个行盒子，而多个行盒子组成一个页盒子。



1.5.1 左右盒子

左右盒子是最简单的盒子，它包括 `\mbox` 和 `\fbox`。例如：

正文\mbox{盒子}正文\fbox{盒子}正文

正文盒子正文

盒子

正文



可以看出两者的区别在于后者带有边框。

对应的还有另外两种左右盒子 `\makebox` 和 `\framebox`，它们提供更多的选项，比如可以指定盒子的宽度：

```
正文\makebox[8em]{盒子}正文\\
正文\framebox[8em]{盒子}正文
```

```
正文      盒子      正文
正文[      盒子      ]正文
```

而且还可以指定盒子中文本的对齐方式，例如：

```
正文\makebox[10em][l]{靠左对齐}正文\\
正文\framebox[10em][l]{靠左对齐}正文
```

```
正文靠左对齐      正文
正文[靠左对齐      ]正文
```

```
正文\makebox[10em][c]{居中对齐}正文\\
正文\framebox[10em][c]{居中对齐}正文
```

```
正文      居中对齐      正文
正文[      居中对齐      ]正文
```

```
正文\makebox[10em][r]{靠右对齐}正文\\
正文\framebox[10em][r]{靠右对齐}正文
```

```
正文      靠右对齐正文
正文[      靠右对齐]正文
```

```
正文\makebox[10em][s]{分散对齐}正文\\
正文\framebox[10em][s]{分散对齐}正文
```

```
正文分    散    对    齐正文
正文[分    散    对    齐]正文
```

```
正文\makebox[10em][s]{A Lazy Dog}正文\\
正文\framebox[10em][s]{A Lazy Dog}正文
```

```
正文A      Lazy      Dog正文
正文[A      Lazy      Dog]正文
```

其中 `l`、`c`、`r`、`s` 分别表示靠左、居中、靠右和分散对齐。使用分散对齐选项时，对中文以汉字为单位分散，而对于西文则以单词为单位分散。

在定义 `\makebox` 和 `\framebox` 的宽度时，我们可以用 `\width`、`\height`、`\depth` 和 `\totalheight` 这些长度量，它们分别表示盒子内容的宽度、高度、深度和总高度（即高度和深度之和）。比如下面例子设定盒子的宽度等于内容宽度的三倍：

```
正文\framebox[3\width]{盒子}正文
```

```
正文[      盒子      ]正文
```

对于带框盒子 `\fbox` 和 `\framebox`，我们可以用 `\fboxsep` 调整边框和内容之间的距离，用 `\fboxrule` 调整框线的宽度。两者的默认值分别为 `3pt` 和 `0.4pt`。例如：

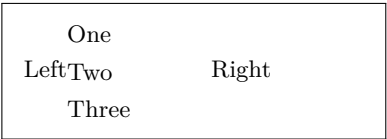
```
\setlength{\fboxsep}{8pt}
\setlength{\fboxrule}{1pt}
正文\fbox{盒子}正文
```




### 1.5.2 段落盒子

左右盒子中不能包含多行文本或者段落。此时我们可以改用 `\parbox` 命令或者 `minipage` 环境。例如：

```
Left\parbox{6em}{One\\ Two\\ Three}Right
```



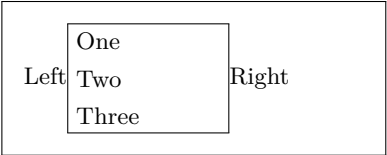
```
Left\begin{minipage}{6em}
One\\ Two\\ Three
\end{minipage}Right
```



其中必须指定盒子宽度。两者的用法和参数都一样，只是在 `\parbox` 中不能包含某些命令或环境，而 `minipage` 中的内容几乎没有限制。

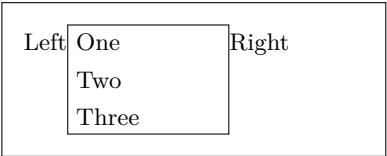
`\parbox` 和 `minipage` 都不含边框，但我们可以将它们放在 `\fbox` 中得到边框，例如：

```
Left\fbox{\begin{minipage}{6em}
One\\ Two\\ Three
\end{minipage}}Right
```

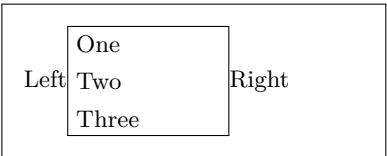


段落盒子还有三个可选参数，其中第一个用于设定段落盒子和两边内容的对齐方式。例如：

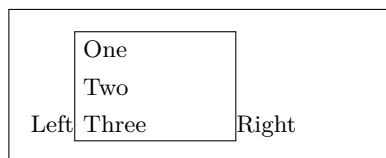
```
Left\fbox{\begin{minipage}[t]{6em}
One\\ Two\\ Three
\end{minipage}}Right
```



```
Left\fbox{\begin{minipage}[c]{6em}
One\\ Two\\ Three
\end{minipage}}Right
```



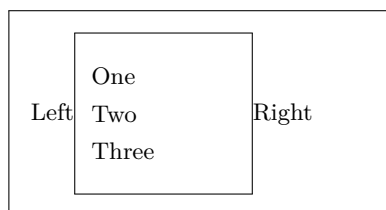
```
Left\fbbox{\begin{minipage}[b]{6em}
One\\ Two\\ Three
\end{minipage}}Right
```



其中，默认选项 `c` 表示段落盒子的中线和外部内容的基线对齐，而选项 `t` 表示盒子顶行的基线和外部内容的基线对齐，选项 `b` 表示盒子底行的基线和外部内容的基线对齐。

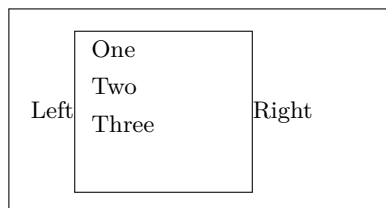
第二个可选参数用于设定段落盒子的高度。例如：

```
Left\fbbox{
\begin{minipage}[c][6em]{6em}
One\\ Two\\ Three
\end{minipage}
}Right
```

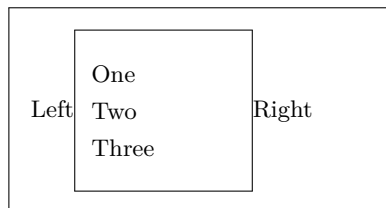


第三个可选参数用于设定段落盒子内部的纵向对齐方式。例如：

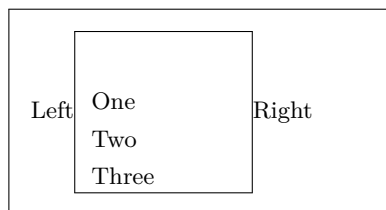
```
Left\fbbox{
\begin{minipage}[c][6em][t]{6em}
One\\ Two\\ Three
\end{minipage}
}Right
```



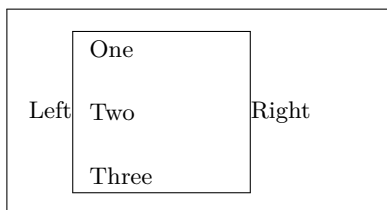
```
Left\fbbox{
\begin{minipage}[c][6em][c]{6em}
One\\ Two\\ Three
\end{minipage}
}Right
```



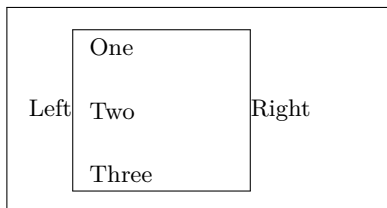
```
Left\fbbox{
\begin{minipage}[c][6em][b]{6em}
One\\ Two\\ Three
\end{minipage}
}Right
```



```
Left\fbbox{
  \begin{minipage}[c][6em][s]{6em}
    One\vfill Two\vfill Three
  \end{minipage}
}Right
```



```
Left\fbbox{
  \begin{minipage}[c][6em][s]{6em}
    \setlength{\baselineskip}{10pt plus 2pt}
    One\\ Two\\ Three
  \end{minipage}
}Right
```

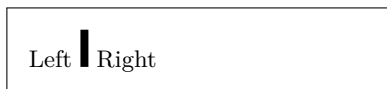


其中默认值 `c` 表示居中对齐，`t` 和 `b` 分别表示顶部对齐和底部对齐，而 `s` 表示分散对齐。在使用分散对齐时，内容中需要有竖直的弹性长度。上面的例子中，分别通过直接使用 `\vfill` 命令和设定弹性行距 `\baselineskip` 得到弹性长度。

### 1.5.3 标尺盒子

$\text{\LaTeX}$  中还提供了 `\rule` 命令，用于生成标尺盒子，即一个不含内容的实心矩形盒子。例如：

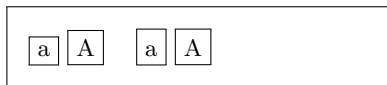
```
Left \rule{3pt}{14pt} Right
```



其中 `\rule` 命令的两个参数分别表示宽度和高度。

宽度为零的标尺盒子虽然无法看到，但可以起到支撑的作用，我们称之为支架（strut）。例如：

```
\fbbox{a} \fbbox{A} \quad
\fbbox{a\rule{0pt}{7pt}}
\fbbox{A\rule{0pt}{7pt}}
```



在这个例子中，由于 `a` 和 `A` 的高度不一，左边两个盒子的高度不一样，但加入支架后右边两个盒子的高度就相同了。

标尺盒子默认与该行的基线平行，通过设定 `\rule` 命令的可选参数，可以升高或降低其水平位置。例如：

```
默认\rule{3pt}{14pt}默认
上升\rule{5pt}{3pt}{14pt}上升
下降\rule{-5pt}{3pt}{14pt}下降
```

默认默认上升上升下降下降

### 1.5.4 幻影盒子

L<sup>A</sup>T<sub>E</sub>X 中还有一种幻影盒子，它们的内容看不到，但内容所占据的空间保持不变。幻影盒子可以用 `\phantom` 命令得到，例如：

```
正文\phantom{幻影}正文
```

正文 正文

类似地还有 `\hphantom` 和 `\vphantom` 命令。`\hphantom` 用于生成水平幻影，其宽度等于内容的宽度，而高度和深度为零。`\vphantom` 用于生成竖直幻影，其高度和深度分别等于内容的高度和深度，而宽度为零。例如：

```
正文正文\\
正文\hphantom{水平幻影}正文\\
正文\vphantom{\Huge 竖直幻影}正文
```

正文正文  
正文 正文  
正文正文

在这个例子中，竖直幻影的高度较大，撑开了第二行和第三行的间距。

### 1.5.5 盒子变换

利用 `\raisebox` 可以升高或者降低盒子的位置。例如：

```
正文 \raisebox{6pt}{上升} 正文\\
正文 \raisebox{-6pt}{\fbox{下降}} 正文
```

正文 上升 正文  
正文 下降 正文

```
L\raisebox{1pt}{\tiny A}%
T\raisebox{-2pt}{E}X
```

L<sup>A</sup>T<sub>E</sub>X

我们也可以在 `\raisebox` 命令中直接指定盒子的最终高度和深度。例如：

```
正文正文正文\\
正文
\raisebox{6pt}[30pt][9pt]{\fbox{上升}}
正文
```

正文正文正文  
正文 上升 正文

其中两个可选参数分别表示所需的高度和深度，深度也可以不指定。

利用 `\smash` 命令，可以将已有的盒子拍扁，保持宽度不变，但将高度和深度变为零。例如：

```
正文正文\\
正文{\huge 盒子}正文\\
正文\smash{\huge 盒子}正文
```

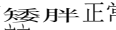

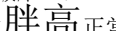
正文正文  
正文  正文  
正文  正文

其中第三行的“盒子”的高度已经变为零，所以  $\text{\TeX}$  排版时不会自动增加第二行和第三行的间距，使得两行部分重叠。

在 `graphics` 和 `graphicx` 宏包中还提供了这些盒子变换命令：`\resizebox`、`\scalebox`、`\reflectbox` 和 `\rotatebox`。

利用 `\resizebox` 命令，可以对盒子进行伸缩变换，例如：

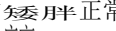


```
正常\resizebox{30pt}{5pt}{矮胖}正常\\
正常\resizebox{11pt}{13pt}{瘦高}正常\\
正常\resizebox{30pt}{!}{胖高}正常
```

正常  正常  
正常  正常  
正常  正常

该命令的两个参数分别表示给该盒子指定的宽度和总高度。如果其中一个参数取为 `!`，则表示保持宽高比例不变且按照另一参数进行伸缩。

利用 `\scalebox` 命令，同样可以对盒子进行伸缩变换，例如：



```
正常\scalebox{1.6}[0.7]{矮胖}正常\\
正常\scalebox{0.7}[1.6]{瘦高}正常\\
正常\scalebox{1.6}{胖高}正常
```

正常  正常  
正常  正常  
正常  正常

该命令的两个参数分别表示水平和竖直伸缩因子。如果省略竖直伸缩因子，则表示它和水平伸缩因子相同。

`\scalebox` 命令的伸缩因子也可以是负数，水平伸缩因子为负数表示伸缩后作水平翻转，而竖直伸缩因子为负数表示伸缩后作竖直翻转。例如：

```
正常\scalebox{-1.6}[0.7]{矮胖}正常\\
正常\scalebox{0.7}[-1.6]{瘦高}正常
```

正常  正常  
正常  正常

特别地，水平和竖直伸缩因子分别为 `-1` 和 `1` 时就表示仅作水平翻转，这等同于 `\reflectbox` 命令。例如：

正常\scalebox{-1}[1]{翻转}正常\\  
正常\reflectbox{翻转}正常

正常𐀀正常  
正常𐀀正常

X\raisebox{-2pt}{\reflectbox{E}}%  
T\raisebox{-2pt}{E}X

X<sub>E</sub>T<sub>E</sub>X

利用 \rotatebox 命令，可以对盒子进行旋转变换，例如：

正常\rotatebox{45}{逆时针}正常\\  
正常\rotatebox{-30}{顺时针}正常

正常逆时针正常

正常顺时针正常

其中的参数指明旋转角度，大于零表示逆时针旋转，小于零表示顺时针旋转。

1.6 插入表格

插入表格经常使用的是 tabular 环境，这个环境是 LaTeX 中预先定义好的。

1.6.1 基本表格

我们从下面的例子来看看如何用 tabular 环境插入表格：

```
\begin{tabular}{|l|c|r|}  
\hline  
左列 & 中列 & 右列 \\  
\hline  
第二行 & 第二行 & 第二行 \\  
\hline  
左三 & 中三 & 右三 \\  
\hline  
\end{tabular}
```

左列	中列	右列
第二行	第二行	第二行
左三	中三	右三

我们来解释这个例子。首先，tabular 环境的参数 |l|c|r| 指明了各列的对齐方式，l、c 和 r 分别表示左对齐、居中对齐和右对齐。中间的竖线 | 指明各列之间有竖线分隔，如果在某些地方不需要竖线，去掉相应位置的 | 即可。

表格各行的元素之间用 `&` 号分隔，两行内容用 `\\` 分隔。`\hline` 表示两行之间的横线；你可以用连续两个 `\hline` 得到双横线，或者去掉 `\hline` 以不显示该横线。

如果需要在某个单元格中填写多行内容，不能直接用 `\\` 或 `\newline` 命令，而应该将它们放在一个盒子里面（比如 `\parbox` 盒子）。

1.6.2 跨列表格

复杂的表格经常需要跨行和跨列，在 `tabular` 环境中，我们可以用命令 `\multicolumn` 得到跨列表格，而跨行表格需要使用 `multirow` 宏包，我们暂不介绍。

现在来看看跨列表格的例子：

```
\begin{tabular}{|c|c|c|}
\hline
第一行 & 第一行 & 第一行 \\
\hline
\multicolumn{2}{|c|}{跨列} & 第二行\\
\hline
第三行 & 第三行 & 第三行 \\
\hline
\end{tabular}
```

第一行	第一行	第一行
跨列		第二行
第三行	第三行	第三行

上面的 `\multicolumn` 命令的第一个参数指明要横跨的列数，第二个参数指明对齐和边框线，第三个参数指明该单元格的内容。

1.6.3 浮动表格

前面所说的插入表格的例子中，表格是在 `tabular` 环境对应的位置排版出来的。如果表格高度大于当前页剩余高度，表格就会被放置到下一页中，造成这一页下部留出很大空白。

大部分时候我们并不需要严格限定表格出现的位置，而只要求表格在该段正文的附近出现。此时，我们可以用 `table` 浮动环境来达到自动调整位置的效果。例如下面的代码得到的是自动浮动的表 1.1：



```

\begin{table}[htbp!]
\centering
\begin{tabular}{|l|c|r|}
\hline
第一行 & 第一行 & 第一行 & 第一行 \\
\hline
第二行 & 第二行 & 第二行 & 第二行 \\
\hline
\end{tabular}
\caption{浮动表格例子}
\label{tab:float}
\end{table}

```

第一行	第一行	第一行	第一行
第二行	第二行	第二行	第二行

表 1.1: 浮动表格例子

其中的可选参数里，**h** (here, 当前位置)、**t** (top, 页面顶部)、**b** (bottom, 页面底部)、**p** (page, 单独一页) 表明允许将表格放置在哪些位置，而 **!** 表示不管某些浮动的限制。用 **table** 浮动环境，还可以用 **\caption** 命令指明表格的名称，并得到表格的自动编号。

在 **\caption** 后面用 **\label** 命令，可以给浮动表格加上标签。在文档其它地方可以用 **\ref** 命令引用该表格。

## 1.7 插入图片

在 L<sup>A</sup>T<sub>E</sub>X 文档中插入现有的图形，可以使用 **graphics** 或 **graphicx** 宏包，其中 **graphicx** 宏包是对 **graphics** 宏包的改进。我们这里只介绍 **graphicx** 宏包。这一节中的例子都是在载入 **graphicx** 宏包之后才能使用的：

```
\usepackage{graphicx}
```

### 1.7.1 图文并排

首先我们看看小图片的插入。用 **\includegraphics[选项]{图形文件}** 命令来插入图形。例如，如下的代码



向日葵是一种花。向日葵是一种花。向日葵是一种花。向日葵是一种花。



向日葵是一种花。向日葵是一种花。向日葵是一种花。向日葵是一种花。

图片的伸缩因子需要计算有些麻烦，如果将它设为页芯宽度 `\textwidth` 的某个因子就简单多了。例如下面的代码

向日葵是一种花。向日葵是一种花。向日葵是一种花。向日葵是一种花。

```
\begin{center}
\includegraphics[width=0.4\textwidth]{picture/kuihua.jpg}
\end{center}
```

向日葵是一种花。向日葵是一种花。向日葵是一种花。向日葵是一种花。

可以得到下面的结果：

向日葵是一种花。向日葵是一种花。向日葵是一种花。向日葵是一种花。



向日葵是一种花。向日葵是一种花。向日葵是一种花。向日葵是一种花。

### 1.7.3 浮动图片

和插入表格的情形类似，有时候我们也需要自动调整图片的位置。此时，我们可以用 `figure` 浮动环境 来达到这个效果。例如下面的代码得到的是自动浮动的图 1.1:

```
\begin{figure}[htbp!]  
\centering  
\includegraphics[scale=0.3]{picture/kuihua.jpg}  
\caption{向日葵照片}  
\label{fig:float}  
\end{figure}
```



图 1.1: 向日葵照片

其中的可选参数和浮动表格的 `table` 环境的一样,而且同样可以用 `\caption` 命令指明图片的名称,并得到图片的自动编号。

在 `\caption` 后面用 `\label` 命令,可以给浮动图片加上标签。在文档其它地方可以用 `\ref` 命令引用该图片。

# 第二章 格式调整

## 2.1 空白调整

### 2.1.1 长度单位

在 LaTeX 中会碰到各种长度单位，这里统一介绍如下。

长度单位	换算关系
cm（厘米）	1cm=10mm
pt（点）	1pt=0.351mm
bp（大点）	1bp=0.353mm
pc（pica）	1pc=12pt=4.218mm
in（英寸）	1in=72.27pt=72bp=25.4mm
sp（scaled point）	TeX 系统最小长度单位（65536sp=1pt）
ex	当前字体中 x 的高度
em	当前字体中 M 的宽度

表 2.1: 长度单位

### 2.1.2 行距调整

在 TeX 中，默认的行距通常是字体大小的 1.2 倍。如果字体大小为 10pt，默认行距 `\baselineskip` 将为 12pt。若要调整行距，可以直接修改 `\baselineskip` 的值。例如下面命令将调整行距为双倍行距：

```
\setlength{\baselineskip}{24pt}
```

其中，在 L<sup>A</sup>T<sub>E</sub>X 中 `\setlength` 命令用于修改各种长度的值。

在 L<sup>A</sup>T<sub>E</sub>X 中调整行距也可以通过修改其伸展因子 `\baselinestretch`。修改后的新行距将等于默认行距乘以伸展因子。例如下面命令将调整行距为双倍行距：

```
\renewcommand{\baselinestretch}{2}\selectfont
```

```
\linespread{2}\selectfont
```

上面两种修改伸展因子的方法是同样的，只是第二种方法更简单点。注意修改伸展因子后不会立即生效，在下次设定字体大小时才能生效，因此我们在后面加上了 `\selectfont` 命令。如果放在导言区则无需此命令，因为文档开始时自然会设定字体大小。

在 L<sup>A</sup>T<sub>E</sub>X 中修改 `\baselineskip` 和 `\baselinestretch` 都可以改变行距。若同时修改两者，并不会叠加行距，而是以后一次修改为准。这是因为每次设定字体大小时 L<sup>A</sup>T<sub>E</sub>X 都保存了默认行距，而修改 `\baselinestretch` 时使用默认行距，而不是当前 `\baselineskip` 值，来计算新行距。

在一个段落的各行中，行距 `\baselineskip` 的值保持相同。若在同一个段落中多次修改行距，就只有最后一次改动生效。

对于双倍行距，有些人认为它应该指行距等于字体大小的两倍。即如果字体大小为 10pt，双倍行距应该为 20pt。为方便设置这种行距倍数，可以用 `setspace` 宏包。

### 2.1.3 弹性长度

除了前面介绍的固定长度，T<sub>E</sub>X 中还有一种弹性长度<sup>1</sup>。行距 `\baselineskip` 实际上就是一个弹性长度。例如我们设置弹性行距如下

```
\setlength{\baselineskip}{12pt plus 2pt minus 1pt}
```

这就表示行距正常值为 12pt，伸长值为 2pt，收缩值为 1pt。实际上，对于弹性长度，T<sub>E</sub>X 排版时所减少的长度总不会超过其收缩值，而所增加的长度却会按照各伸长值的比例平均分配，从而可能超过其伸缩值。在排版长文档时弹性行距会导致各页行距不一，但在排版单页文档时这样却可以让 T<sub>E</sub>X 排满整页。

弹性长度的伸长值和收缩值都是可以省略的。若两者都省略，弹性长度就成为一个固定长度。

---

<sup>1</sup>弹性长度又称为粘连 (glue)。

### 2.1.4 表格行距

表格中的行距默认也等于 `\baselineskip`。但通过修改 `\arraystretch` 因子的值，我们可以单独调整表格的行距。下面例子将表格行距修改为原来的 1.5 倍。

```
\renewcommand\arraystretch{1.5}
```

## 2.2 字体使用

在 LaTeX 中，一个字体有 5 种属性：编码（encoding）、族名（family）、系列（series）、形状（shape）和大小（size）。

### 2.2.1 字体编码

字体的编码根据其中包含的字符种类以及各字符的编号来划分。常见的正文字体的编码有 OT1、T1 和 EU1 等，而数学字体的编码有 OML、OMS 和 OMX 等。

### 2.2.2 字体族名

字体的族名有许多种，一般把它们分为三大类：

- 罗马字体：又称衬线字体，字符笔画的起始处有装饰；例如 `cmr`、`ecr` 和 `lmr` 字体。
- 无衬线字体：又称等线字体，字符笔画的起始处无装饰；例如 `cmss`、`ecss` 和 `lmss` 字体。
- 打字机字体：又称等宽字体，每个字符的宽度都相同；例如 `cmtt`、`ectt` 和 `lmtt` 字体。

我们可以用下面的命令方式来改变字体的族名：

```
\textrm{Roman Family} \\
\textsf{Sans Serif Family} \\
\texttt{Typewriter Family}
```

Roman Family Sans Serif Family Typewriter Family
--

或者用下面的声明方式来改变字体的族名：

```
{\rmfamily Roman Family} \\
{\sffamily Sans Serif Family} \\
{\ttfamily Typewriter Family}
```

Roman Family  
Sans Serif Family  
Typewriter Family

### 2.2.3 字体系列

字体系列根据字体的粗细和宽度来划分。经常用到的如下这两种（正常和粗体）：

```
\textmd{Medium Series} \\
\textbf{Boldface Series} \\
\\
{\mdseries Medium Series} \\
{\bfseries Boldface Series}
```

Medium Series  
**Boldface Series**  
  
Medium Series  
**Boldface Series**

### 2.2.4 字体形状

字体形状主要有这些：直立，斜体，伪斜体和小型大写。可以用下面的命令或声明来改变：

```
\textup{Upright Shape} \\
\textit{Italic Shape} \\
\textsl{Slanted Shape} \\
\textsc{Small Caps Shape} \\
\\
{\upshape Upright Shape} \\
{\itshape Italic Shape} \\
{\slshape Slanted Shape} \\
{\scshape Small Caps Shape}
```

Upright Shape  
*Italic Shape*  
*Slanted Shape*  
SMALL CAPS SHAPE  
  
Upright Shape  
*Italic Shape*  
*Slanted Shape*  
SMALL CAPS SHAPE

对字体的族名、系列和形状这三种样式作了修改之后，可以用 `\textnormal` 命令或者 `\normalfont` 声明来使用默认字体样式。例如：

```
\textit{\textbf{\textsf{Fancy Text}}}\ \\
\textnormal{Normal Text} \\
\\
{\itshape\bfseries\sffamily Fancy Text} \\
{\normalfont Normal Text}
```

***Fancy Text***  
Normal Text  
  
***Fancy Text***  
Normal Text



2.2.5 字体大小

字体命令	排版效果	正常 10pt	正常 11pt	正常 12pt
<code>\tiny</code>	hello	5pt	6pt	6pt
<code>\scriptsize</code>	hello	7pt	8pt	8pt
<code>\footnotesize</code>	hello	8pt	9pt	10pt
<code>\small</code>	hello	9pt	10pt	11pt
<code>\normalsize</code>	hello	10pt	11pt	12pt
<code>\large</code>	hello	12pt	12pt	14pt
<code>\Large</code>	hello	14pt	14pt	17pt
<code>\LARGE</code>	hello	17pt	17pt	20pt
<code>\huge</code>	hello	20pt	20pt	25pt
<code>\Huge</code>	hello	25pt	25pt	25pt

表 2.2: 字体的各种大小

在 LaTeX 中可以用各种命令来改变文本字体的大小，它的实际大小和文档类的正常字体大小（即 `\normalsize` 的大小）设置有关，详细情形见表 2.2:

2.3 页面大小布局

前面已经说明，直接在文档类中设定页面大小时，纸张大小不会随着变化。因此我们推荐用 `geometry` 宏包 来设定页面大小，比如本书的 B5 页面大小是这样设置的：

```
\usepackage[b5paper]{geometry}
```

接着我们来说说如何定制页面的布局，比如正文区域的宽度和高度，和各个边距的大小。LaTeX 中提供了各种命令来定制页面布局，但是非常难用。因此，我们同样推荐用 `geometry` 宏包来调整页面布局。例如本书的页面布局就是用如下的代码设定的：

```
\usepackage[b5paper,text={125mm,195mm},centering]{geometry}
```

其中的 `text={width,height}` 选项指明了页面正文区域的宽度和高度大小，而后面的 `centering` 选项表示将正文区域自动居中（即上下边距相等，而且

左右边距也相等)。

上面的代码也可以分成两行来写，如下：

```
\usepackage{geometry}  
\geometry{b5paper,text={125mm,195mm},centering}
```

如果你想直观地观察设置好的页面布局，可以在 `geometry` 宏包的选项中再加上 `showframe` 一项。这样 `geometry` 宏包将在文档的第一页画出页面的布局。

## 2.4 目录书签

### 2.4.1 添加目录项

利用 `\addcontentsline` 命令，我们可以在目录中添加一项。例如：

```
\addcontentsline{toc}{section}{章节描述}  
\addcontentsline{lof}{figure}{插图描述}  
\addcontentsline{lot}{table}{表格描述}
```

其中，`\addcontentsline` 命令的第一个参数表示添加到哪个目录，`toc`、`lof`、`lot` 分别表示章节目录、插图目录和表格目录。第二个参数表示该项的类型，对于章节目录可以是 `chapter`、`section`、`subsection` 等，而对于插图目录或表格目录只能是 `figure` 或 `table`。第三个参数表示该项在目录中的文本。

对于章节目录， $\text{\TeX}$  程序第一次编译文档时将生成 `.toc` 临时文件，而第二次编译时将从读取 `.toc` 文件内容并生成目录。插图目录和表格目录的情形类似。利用 `tocloft` 宏包，我们还可以生成新的目录，比如例子目录、练习目录等。

利用 `\addcontentsline` 命令，我们可以把“参考文献”和“索引”也添加到目录中：

```
\addcontentsline{toc}{chapter}{\bibname}  
\begin{thebibliography}{123456}  
.....  
\end{thebibliography}
```

```
\addcontentsline{toc}{chapter}{\indexname}
\printindex
```

但参考文献和索引一般都是在新一页开始的，因此这样得到的目录页码是上一页的。在前面加上 `\clearpage` 或者 `\cleardoublepage`，我们可以得到正确的页码。例如：

```
\clearpage
\addcontentsline{toc}{chapter}{\indexname}
\printindex
```

即便显示的页码正确了，此时在 PDF 文档中点击该目录项，跳转到的页码还是错误的。在 `hyperref` 宏包中，有 `\phantomsection` 命令用于处理此问题。例如：

```
\clearpage
\phantomsection
\addcontentsline{toc}{chapter}{\indexname}
\printindex
```

如果仅要在章节目录添加“目录”，“参考文献”和“索引”这几项，用 `tocbibind` 宏包可能更加方便。

### 2.4.2 添加书签项

利用 `\addcontentsline` 命令添加的目录项，同时也会在 PDF 书签中显示。有时候，例如对于“目录”，我们只希望在 PDF 中添加书签项，而不出现现在文档目录中。此时可以用 `hyperref` 宏包提供的 `\pdfbookmark` 命令。例如：

```
\clearpage
\pdfbookmark{全书目录}{Contents}
```

和添加目录项的做法一样，`\clearpage` 命令用于保证得到正确的页码。

`\pdfbookmark` 命令的完整参数如下：

```
\pdfbookmark[级别]{显示文本}{名称}
```

其中，可选参数“级别”默认为 0，“名称”可以自己选择，但不能互相冲突。

在 `hyperref` 宏包的文档中，也推荐用 `bookmark` 宏包来增添书签项。`bookmark` 宏包的优点在于大多数时候只需要编译一次，而且书签的颜色和字体也可以定制。

### 2.4.3 替换书签项

如果章节名称中包含复杂的数学字符，在 PDF 文档的书签中将无法显示出来。我们看下面的例子：

```
\section{$\Gamma$ 函数}
\section{$L^2$ 空间}
```

这个例子编译后，其中的  $\Gamma$  将不会在书签中显示，而  $L^2$  将显示为 L2。`hyperref` 宏包提供了 `\texorpdfstring` 命令，用于替换书签的特殊字符。例如：

```
\section{\texorpdfstring{$\Gamma$ 函数}{\ 函数}}
\section{\texorpdfstring{$L^2$ 空间}{L\texttt{two}superior\ 空间}}
```

在这个例子中，我们用文本形式的  $\Gamma$  和  $^2$  分别替换数学中的  $\Gamma$  和上标 2。

文本形式的上标数字，`textcomp` 宏包中只包含 `\textonesuperior`<sup>(1)</sup>、`\texttt{two}superior`<sup>(2)</sup> 和 `\texttt{three}superior`<sup>(3)</sup> 这三个。最新的 `hyperref` 和 `xeCJK` 宏包完整定义了从 0 到 9 的文本形式上下标数字，并且部分定义了文本形式的下标字母<sup>2</sup>。例如 `\texttt{four}superior`<sup>(4)</sup>、`\texttt{five}inferior`<sub>(5)</sub>、`\texttt{e}inferior`<sub>(e)</sub> 等等。如果使用 `xelatex` 编译，设置合适的字体之后，直接把这些 Unicode 的文本上下标字符粘贴到  $\text{\LaTeX}$  代码中，也是可行的。

### 2.4.4 目录页格式

要调整章节标题在目录页中的格式，可以用 `titletoc` 宏包。该宏包的基本命令参数如下<sup>3</sup>：

```
\titlecontents{标题层次}[左间距]{整体格式}
               {标题序号}{标题内容}{指引线和页码}[下间距]
```

<sup>2</sup> 文本形式的下标字母不全，是否是因为在 Unicode 中有些字符还未收入，有待查证。

<sup>3</sup> Javier Bezos, The titlesec and titletoc Packages, 2007. URL <http://mirrors.ctan.org/macros/latex/contrib/titlesec/titlesec.pdf> (部分的中文翻译在这里: <http://bbs.ctex.org/viewthread.php?tid=52126>)

其中“标题层次”参数可以取为 `part`、`chapter`、`section` 等标题名以及 `figure` 和 `table` 浮动图表名。“左间距”指的是“标题内容与页芯左侧的距离。因为“标题序号”一般在“标题内容”的左侧，所以除非标题居中放置或者标题无序号，“左间距”一般需要取大于 0 的值。实际上，该参数总是不可省略的。

我们通过例子来看看其它参数的使用。比如我们要将章（`chapter`）标题用大号粗体居中放置，同时要保留序号，则可以用下面的命令达到：

```
\titlecontents{chapter}
  [0em]
  {\filcenter\Large\bfseries}
  {\contentslabel{3em}}
  {}
  {}
```

其中“标题序号”参数的 `\contentslabel` 命令指代该部分的序号内容，如果不写上就没有序号。而后面的 `3em` 表示“标题序号”左侧和“标题内容”左侧的距离，这个长度值一般要大于 0，否则两者就重合了。

接下来看看，假如我们需要将节（`section`）标题去掉序号，在离页芯左侧 `4em` 处对齐，并且用居中的点（`\cdot`）来画出指引线，可以用下面的命令达到：

```
\titlecontents{section}
  [4em]
  {}
  {}
  {}
  {\titlerule*[1em]{${\cdot}}\contentspage}
```

其中 `\titlerule*` 命令用于画指引线，`1em` 表明指引线的各个点的距离。而 `\contentpage` 命令表示页码。

如果用 `ctex` 宏包来撰写中文文档，需要先载入 `titletoc` 宏包在载入 `ctex` 宏包，否则中文设置将会覆盖掉。正确的例子如下：

```
\documentclass{article}
\usepackage{titletoc}
\usepackage{ctexcap}
\begin{document}
...
\end{document}
```

# 第三章 数学公式

## 3.1 两种公式

在 L<sup>A</sup>T<sub>E</sub>X 中，数学公式有两种：即行内公式（inline formula）和行间公式（displayed formula）。行内公式和正文在同一行中显示，可以用下面三种方式来表示：

From \$ a+b>c \$, we have ...	From $a + b > c$ , we have ...
From \(\ a+b>c \), we have...	From $a + b > c$ , we have...
From \begin{math}a+b>c\end{math}, we have	From $a + b > c$ , we have

而行间公式在单独一行居中显示，可以用如下三种不同的方法表示：

Since \$\$ x^n + y^n = z^n, \$\$ we have ...	Since $x^n + y^n = z^n,$ we have ...
Since \[ x^n + y^n = z^n, \ we have...	Since $x^n + y^n = z^n,$ we have...
Since \begin{displaymath} x^n + y^n = z^n, \end{displaymath} we have...	Since $x^n + y^n = z^n,$ we have...

一般地, 对于行内公式, 我们常用简短的  $\$...\$$  形式。而对于行间公式, 我们推荐使用  $\backslash[...\backslash]$  形式而不是  $$$...$$$  形式<sup>1</sup>。

用  $$$...$$$  形式来输入行间公式, 除了输入麻烦外还有其它缺点 (这里忽略不提)。但是它也有一个特有的好处, 就是可以用  $\backslasheqno$  命令指定公式的编号, 比如

```
 $\$x_1+y_1>z_1 \backslasheqno{(1)}\$$ 
```

$$x_1 + y_1 > z_1 \quad (1)$$

另外, 我们可以用 `equation` 环境来得到自动编号的行间公式。例如:

```
 $\begin{equation}$   
 $x^n+y^n=z^n$   
 $\end{equation}$ 
```

$$x^n + y^n = z^n \quad (3.1)$$

从上面最简单的例子可以看出, 在数学公式里面, 用 `_` 来表示下标, 而用 `^` 来表示上标 (或幂次)。另外, 公式里的所有空格都是被忽略的, LaTeX 排版程序会自动给出合适的空距。

## 3.2 各种字母

在数学公式里面, 各种希腊字母可以根据它们的英文名对应的命令来得到。例如:

```
 $\backslash[ \backslashalpha\backslashbeta\backslashdelta\backslasheta\backslashgamma\backslashiota$   
 $\backslashlambda\backslashkappa\backslashomega\backslashsigma\backslashzeta \backslash]$   
 $\backslash[ \backslashDelta\backslashGamma\backslashLambda\backslashPhi\backslashPi\backslashPsi$   
 $\backslashSigma\backslashTheta\backslashOmega\backslashUpsilon\backslashXi \backslash]$ 
```

$$\alpha\beta\delta\eta\gamma\iota\lambda\kappa\omega\sigma\zeta$$
  
$$\Delta\Gamma\Lambda\Phi\Pi\Psi\Sigma\Theta\Omega\Upsilon\Xi$$

其中, 首字母大写的命令表示大写的希腊字母。

如果需要各种黑板体和花体字母, 可以首先用下面命令

```
 $\usepackage{amssymb}$ 
```

载入 `amssymb` 包, 然后输入  $\backslashmathbb{Z}$  显示黑板体字母  $\mathbb{Z}$ , 输入  $\backslashmathcal{O}$  显示花体字母  $\mathcal{O}$ 。

<sup>1</sup>两种行间公式的区别见: <http://tex.stackexchange.com/q/503/8956>。

### 3.3 数学函数

常见的一些数学函数可以看下面的一些例子：

```
\[
\frac{1}{2}+\frac{1}{3}=\frac{5}{6}
\]
```

$$\frac{1}{2} + \frac{1}{3} = \frac{5}{6}$$

```
\[
\sqrt{2}\cdot\sqrt[4]{2}=\sqrt[4]{8}
\]
```

$$\sqrt{2} \cdot \sqrt[4]{2} = \sqrt[4]{8}$$

```
\[
2\sin x \cos x = \sin 2x
\]
```

$$2 \sin x \cos x = \sin 2x$$

```
\[
\lim_{n\rightarrow\infty}\frac{1}{n} = 0
\]
```

$$\lim_{n \rightarrow \infty} \frac{1}{n} = 0$$

```
\[
\int\frac{1}{x} \, dx = \ln |x| + C
\]
```

$$\int \frac{1}{x} dx = \ln |x| + C$$

### 3.4 配对括号

在数学公式中，如果直接使用括号，得到的公式会比较丑陋，例如：

```
\[
\lim_x(1+\frac{1}{x})^x = \mathrm{e}
\]
```

$$\lim_x (1 + \frac{1}{x})^x = e$$

我们可以用 `\left` 和 `\right` 命令来得到自动调整大小的括号，例如

```
\[
\lim_x\left(1+\frac{1}{x}\right)^x = e
\]
```

$$\lim_x \left(1 + \frac{1}{x}\right)^x = e$$



自动调整大小的括号有时候效果仍然差强人意，这时候你也可以自己指定括号的大小：

```
\[ \Bigg< \bigg\{ \Big[ \big( xyz \big) \Big] \bigg\} \Bigg> \]
```

$$\left\langle \left\{ \left[ (xyz) \right] \right\} \right\rangle$$

注意 { 和 } 是特殊字符，需要用 \{ 和 \} 来表示。

## 3.5 多行公式

要方便地输入多行公式，可以使用美国数学会的 `amsmath` 宏包。本节介绍的这些环境在载入了该宏包后才能使用。

首先来看最简单的多行公式环境，即 `gather` 环境。例如：

```
\begin{gather}
x + y = 5 \\\
2x + 3y = 8
\end{gather}
```

$$\begin{array}{rcl} x + y & = & 5 \quad (3.2) \\ 2x + 3y & = & 8 \quad (3.3) \end{array}$$

其中 `\\` 符号用于分割各行。从这个例子可以看出，在 `gather` 环境的多行公式里面，各行之间是不对齐的。如果要得到对齐的公式，可以用 `align` 环境。例如：

```
\begin{align}
x + y &= 5 \\\
2x + 3y &= 8
\end{align}
```

$$\begin{array}{rcl} x + y & = & 5 \quad (3.4) \\ 2x + 3y & = & 8 \quad (3.5) \end{array}$$

其中 `&` 符号后面的字符就是各行间对齐的位置。

前面两个环境中的多行公式都是自动编号的。如果不要编号，可以用对应的 `gather*` 和 `align*` 环境。例如：

```
\begin{gather*}
x + y = 5 \\\
2x + 3y = 8
\end{gather*}
```

$$\begin{array}{rcl} x + y & = & 5 \\ 2x + 3y & = & 8 \end{array}$$

```
\begin{align*}
x + y &= 5 \\\
2x + 3y &= 8 \\
\end{align*}
```

$$\begin{aligned}x + y &= 5 \\ 2x + 3y &= 8\end{aligned}$$

有时候，一个公式太长需要拆为几行，这种折行公式应该只需要一个编号，因此 `gather` 和 `align` 环境就不适用了。这时候，我们可以用 `\split` 环境。例如：

```
\begin{equation}
\begin{split}
(2+3)\cdot 5 &= 2\cdot 5 + 3\cdot 5 \\\
&= 25 \\
\end{split}
\end{equation}
```

$$\begin{aligned}(2+3) \cdot 5 &= 2 \cdot 5 + 3 \cdot 5 \\ &= 25\end{aligned} \quad (3.6)$$

注意 `split` 环境必须放置在 `equation` 环境中。我们称这种在其它环境内部才能使用的环境为次环境。

最后，`amsmath` 宏包还提供 `gathered` 和 `aligned` 这两个环境。这两个环境也是次环境，必须在数学环境中才能使用。这两者的最大特点在于，它们不会占用整个宽度，因此可以作为整体放置在一个复杂公式里面（这种环境我们称为块环境）。例如：

```
\begin{equation}
\left. \begin{aligned}
x+y &> 5 \\\
y-y &> 11 \\
\end{aligned} \right\} \Rightarrow x^2 - y^2 > 55 \\
\end{equation}
```

$$\left. \begin{aligned}x + y &> 5 \\ y - y &> 11\end{aligned} \right\} \Rightarrow x^2 - y^2 > 55 \quad (3.7)$$

注记：LaTeX 本身也提供了用于排版多行对齐公式的 `eqnarray` 环境，例如：

```
\begin{eqnarray*}
a x + b y &=& u \\\
c x + d y &=& v \\
\end{eqnarray*}
```

$$\begin{aligned}ax + by &= u \\ cx + dy &= v\end{aligned}$$

其中两个 & 号之间的是公式间对齐的位置，用 \\ 隔开各行公式。上面输出的公式是没有编号的，如果需要自动编号，可以将 `eqnarray*` 改为 `eqnarray`。这个 `eqnarray` 环境的有不少问题，已经不再建议使用。<sup>2</sup>

## 3.6 画交换图

在  $\text{\LaTeX}$  中画交换图，可以用 `amscd` 包，或者 `xy-pic` 包。

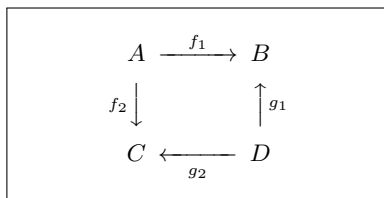
### 3.6.1 用 `amscd` 包画交换图

首先需要在文档的导言区载入 `amscd` 宏包：

```
\usepackage{amscd}
```

然后就可以用类似下面的代码画交换图了：

```
\[
\begin{CD}
A @>f_1>> B \\
@Vf_2VV @AAg_1A \\
C @<<g_2<< D
\end{CD}
\]
```



其中 `@>>>`，`@<<<`，`@AAA`，`@VVV` 分别表示向左，向右，向上，向下的箭头。`amscd` 宏包只能画方型的交换图。

### 3.6.2 用 `xy-pic` 包画交换图

前面介绍了用 `amscd` 包画交换图的方法，现在介绍用 `xy-pic` 包画交换图的方法<sup>3</sup>，这个包的使用更加直观，功能也强大，推荐使用它。

$\text{\LaTeX}$  里已经包含了这个包，只要调用了这个包

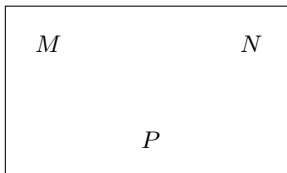
```
\usepackage{all}{xy}
```

<sup>2</sup>Lars Madsen, Avoid `eqnarray!`, The  $\text{\LaTeX}$  Journal, 2006, No.4, <http://tug.org/pracjourn/2006-4/madsen/madsen.pdf>.

<sup>3</sup>K. Rose, XY-pic User's Guide

就可以使用了。xy-pic 的原理是先以矩阵方式画出各个元素，然后画出各个元素之间的箭头。首先用命令画出个矩阵，用 & 隔开各个元素，用 \\ 换行，某些元素也可以空着，例如：

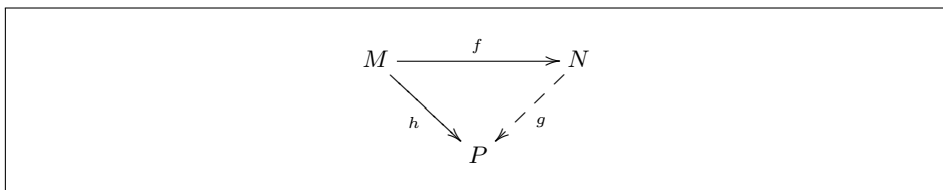
```
\xymatrix{
M & & N \\
& P &
}
```



然后指明要画的箭头，比如下面的代码

```
\xymatrix{
M \ar[rr]^{f}\ar[dr]_{h} & & N \ar@{-->}[dl]^{g}\\
& P &
}
```

得到的交换图是



代码中的 \ar 命令 (arrow) 后方括号里的字符串指明了箭头的方向，u、d、l、r 分别指上下左右，比如 rr 表示指向右边第二个元素，dl 表示指向左边那个元素。

\ar 后的 @{-->} 表示箭头的类型，还有 @{=>}, @{.>}, @{~>}, @{-} 等等类型，不加上这个即使用默认的箭头。

\ar 后的 ^{f} 和 \_{h} 表示箭头上的标记，^ 表示放在箭头前进方向的左侧，\_ 表示放在箭头前进方向的右侧。标记将会放在两个元素的正中间，而使用 ^-{f} 和 \_-{h} 将会使标记放在箭头的正中间。

注意 xy-pic 交换图要放在数学公式中。

### 3.7 公式行距

在多行公式中，行距默认等于 \baselineskip 加上 \jot 长度值。 \jot 默认等于 3pt。

在 `array` 环境，以及由它定义的环境（例如 `amsmath` 的 `vmatrix` 等）中，行距默认等于 `\baselineskip`。但通过修改 `\arraystretch` 因子的值，我们可以调整此类公式的行距。下面例子将其中的行距修改为原来的 1.5 倍。

```
\renewcommand\arraystretch{1.5}
```

实际上，`tabular` 和 `array` 中的行距同样受 `\arraystretch` 因子的影响。

## 第四章 图形绘制

### 4.1 画图语言和画图软件

LaTeX 中的矢量画图语言有 `metapost`、`asymptote` 和 `pgf/tikz`。用这些语言作图，首先得按照它们提供的命令写一个文件，然后编译才得到图形。

**Metapost:** 由 John Hobby 设计的一种绘图语言，从 `metafont` 演化而来的，历史悠久。文件后缀为 `.mp`，用 `mpost.exe` 编译得到图片。

**Asymptote:** 新兴的作图语言，类似于 C 语言的语法。文件后缀为 `.asy`，用 `asy.exe` 编译得到图片。

**PGF/TikZ:** `beamer` 包作者的另一作品，与前两者的区别在于它可以直接在 `tex` 文件中编写，然后用 `pdflatex` 编译。

LaTeX 中的矢量画图软件包括 `TpX`、`Ipe` 和 `Inkscape`，画图软件相对于画图语言来说更加简单直观，但是功能上会有所限制。

**TpX:** 用 Delphi 编写，简单小巧。输出的结果是一个形如 `abc.TpX` 文件和一些图片，只要 `\input{abc.TpX}` 即可。

**Ipe:** 用 Qt 库编写，强大且支持扩展。输出结果是各种格式（包括 `eps` 和 `pdf`）的图片。

**Inkscape:** 用 Gtk+ 编写，非常强大，可以输出为 `svg` 文件，可惜 Gtk+ 编写的程序在 Windows 下不好用。

### 4.2 PGF/TikZ 绘图

要用 PGF/TikZ 绘图，首先需要在导言区载入 `tikz` 宏包。然后就可以用下面的 `tikzpicture` 环境来绘图了<sup>1 2</sup>：

---

<sup>1</sup><http://mirror.ctan.org/graphics/pgf/base/doc/generic/pgf/pgfmanual.pdf>

<sup>2</sup><http://www.texample.net/tikz/examples/>

```
\begin{tikzpicture}
...
\end{tikzpicture}
```

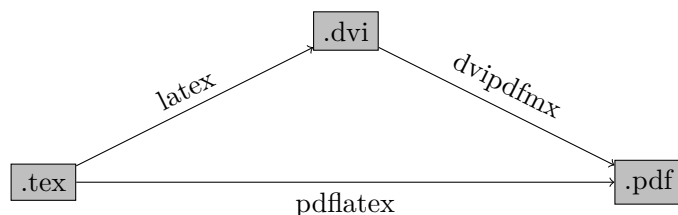
首先需要注意的是，在 pgf/tikz 中，默认的长度单位是厘米（cm）。

### 4.2.1 画流程图

要用 pgf/tikz 画流程图，比如本书最前面的编译流程图，可以用类似下面的代码：

```
\tikzset{
  box/.style = {rectangle, draw=black, fill=lightgray}
}
\begin{tikzpicture}
\node[box] (tex) at(0,0)  {.tex};
\node[box] (pdf) at(8,0)  {.pdf};
\node[box] (dvi) at(4,2)  {.dvi};
\draw[->] (tex) -- node[below]{pdflatex} (pdf);
\draw[->] (tex) -- node[above,sloped]{latex} (dvi);
\draw[->] (dvi) -- node[above,sloped]{dvi2pdf} (pdf);
\end{tikzpicture}
```

结果如下图所示：



这个流程图包含三个类型为 `box` 的节点，在第一行的 `\tikzset` 命令中我们对这种 `box` 类型的节点的样式作了定制。接下来，在 `tikzpicture` 代码中的前面三行在不同位置画出并标记了这三个节点，后面三行对画出并标记了三者之间的指向箭头。

### 4.2.2 函数图像

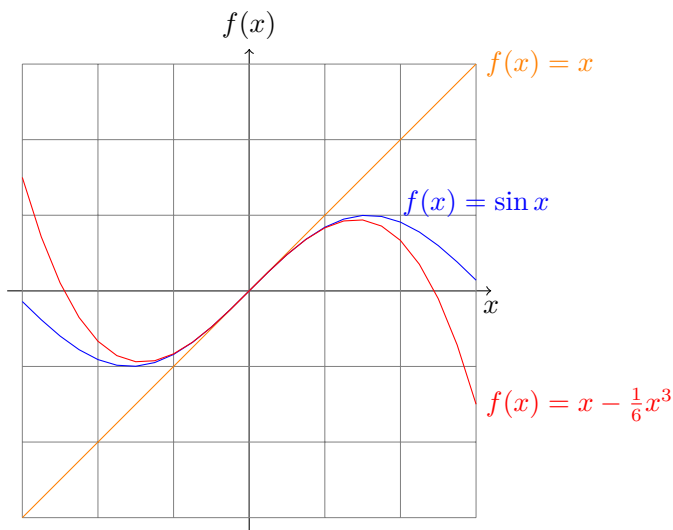
我们再看看怎么画函数图像。下面的代码将绘制几个函数的图形。

```

\begin{tikzpicture}[domain=-3:3]
\draw[->] (-3.2,0) -- (3.2,0) node[below] {$x$};
\draw[->] (0,-3.2) -- (0,3.2) node[above] {$f(x)$};
\draw[very thin,color=gray] (-3,-3) grid (3,3);
\draw[color=orange] plot (\x,\x) node[right] {$f(x)=x$};
\draw[color=blue] plot (\x,{sin(\x r)}) node[above=7mm] {$f(x)=\sin x$};
\draw[color=red] plot(\x,{\x-(1/6)*(\x)^3}) node[right]{$f(x)=x-\frac{1}{6}x^3$};
\end{tikzpicture}

```

输出结果如下：



我们来看看上面的 `tikzpicture` 代码。`tikzpicture` 代码中的 `domain` 选项指明了图形的绘制区域：定义域为  $-3\text{cm}$  到  $3\text{cm}$ 。

前面两行的 `\draw` 命令绘制了两条带箭头的直线作为坐标系，后面的 `node` 参数表示分别在该直线的下方和右边用 LaTeX 代码标记。

第三行代码表示在  $(-3, -3)$  到  $(3, 3)$  的矩形范围内绘制网格，默认的距离为  $1\text{cm}$ 。

后面三行代码用不同颜色绘制了三个函数的图形，其中 `plot` 参数后面紧接着的是  $x$  和  $y$  坐标的表达式（实际上，绘制参数方程曲线也是可以的）。注意其中的 `sin` 函数需要指明长度类型 `r`（表示弧度，默认为角度）。

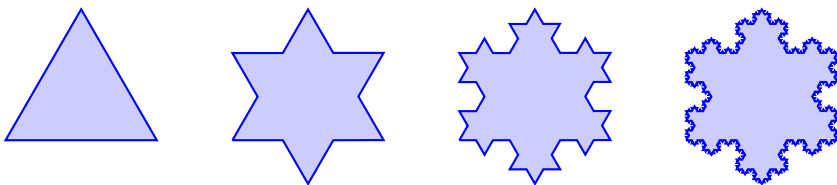
### 4.2.3 几何图形

下面的代码将绘制 Koch 雪花这个分形图形：



```
\usetikzlibrary{decorations.fractals}
\begin{tikzpicture}[decoration=Koch snowflake,draw=blue,fill=blue!20,thick]
\filldraw (0,0) -- ++(60:2) -- ++(-60:2) -- cycle ;
\filldraw decorate{ (3,0) -- ++(60:2) -- ++(-60:2) -- cycle };
\filldraw decorate{ decorate{ (6,0) -- ++(60:2) -- ++(-60:2) -- cycle }};
\filldraw decorate{ decorate{ decorate{ decorate{
(9,0) -- ++(60:2) -- ++(-60:2) -- cycle }}}};
\end{tikzpicture}
```

输出的结果如下：



## 第五章 演示文稿

类似于微软的 PowerPoint 软件，使用 LaTeX 也可以制作 PDF 格式的演示文稿。我们这里用“演示文稿”表示整个文档，而用“幻灯片”表示该文档的每页内容。

用 LaTeX 制作 PDF 演示文稿，有许多文档类或者宏包可以使用：`prospcr`、`powerdot`、`texpower`、`powersem`、`pdfslide`、`pdfscreen`、`beamer` 等等<sup>1</sup>。这里简要介绍如下：

**prospcr**，**ha-prospcr** 和 **powerdot** 依时间先后是同一个系列的演示文档类，所以现在只需要用最新的 `powerdot`。`powerdot` 支持逐步显示，但是只能先用 `latex` 编译得到 `dvi` 文件再依次转换为 `ps` 和 `pdf` 文件，流程过于复杂，所以不推荐使用。

**powersem** 也是一个演示文档类，而 `texpower` 是提供彩色动画和逐步显示功能的宏包，二者捆绑在一起发布。`powersem` 文档类可以用 `pdflatex` 直接编译，很方便。在功能上 `powersem` 属于中等水平。

**pdfscreen** 和 **pdfslide** 这两者都是 C. V. Radhakrishnan 制作的 LaTeX 宏包，在 `article` 文档类中使用。从名字可以看出来二者（只）支持用 `pdflatex` 编译。`pdfscreen` 稍微强大一点，最值得称道的功能是每页的内容能够自动居中（用 `slide` 环境来包含每一页的内容）；而 `pdfslide` 稍微简单一点，得用 `\newpage` 手工分页，而每页内容也不会自动居中。`pdfscreen` 和 `pdfslide` 在功能上是属于初等水平，它们最大问题在于本身不支持逐步显示。逐步显示功能虽然可以用 `ppower4` 作后期处理得到，但是 `ppower4` 是个 java 程序，使用起来麻烦。不过还有上面的 `texpower` 包可用，可是 `texpower` 的 `\pause` 和 `\stepwise` 等命令均不支持在 `pdfscreen` 的 `slide` 环境中使用，所以需要逐步显示的话用 `pdfscreen` 就不大可行了。

---

<sup>1</sup><http://www.ctan.org/tex-archive/help/Catalogue/bytopic.html#present>

而 pdfslide 加 texpower 是最简单的一种 PDF 演示文稿制作方式。(最后, 还有 pdfslide 的改进版 ifmslide 可以使用。)

**beamer** 是功能最强大的演示文档类。它和画图宏包 pgf/tikz 一样都是 Till Tantau 制作的, 二者结合得非常好。beamer 支持 latex 和 pdflatex 编译, 样式繁多, 功能复杂, 使得它成为最流行的 PDF 演示文稿制作方式。

总而言之, 追求简单选 pdfslide 加 texpower, 追求强大用 beamer。接下来我们只介绍 beamer<sup>2</sup>。

## 5.1 最简单例子

我们现在开始来看看, 如何用 beamer 文档类制作 PDF 幻灯片。下面的例子改自 beamer 文档:

```
\documentclass{beamer}

\begin{document}

\title{Introduction to the Beamer Package}
\author{Till Tantau}
\date{\today}

\begin{frame}
\titlepage
\end{frame}

\begin{frame}
\frametitle{Outline}
\tableofcontents
\end{frame}

\section{Other Presentation Packages}

\begin{frame}
\frametitle{Other Packages}
\begin{enumerate}
```

---

<sup>2</sup>Till Tantau & Joseph Wright & Vedran Miletic, The Beamer Class, 2011, <http://mirrors.ctan.org/macros/latex/contrib/beamer/doc/beameruserguide.pdf>.

```
\item Powerdot class
\item Powersem class
\item Pdfscreen package
\end{enumerate}
\end{frame}

\section{Overview of the Beamer Class}

\begin{frame}\frametitle{Beamer Package}
\begin{itemize}
\item Normal LaTeX class.
\item Easy overlays.
\item No external programs needed.
\end{itemize}
\end{frame}

\end{document}
```

看起来非常显然：其中的每个 `frame` 环境表示一张幻灯片，`\frametitle` 表示该幻灯片的标题；而标题页和目录页就是前面两张幻灯片。

在 beamer 文档中，章节命令应该放在各个 `frame` 环境之间，否则在有逐步显示的内容会导致多个编号的问题。

## 5.2 逐步显示

如果需要得到幻灯片动画，首先可以用 `\pause` 命令得到逐步显示。例如：

```
\begin{frame}
Hello \pause World!
\end{frame}
```

而对于列表环境的各项，beamer 中可以用 `<>` 参数控制显示顺序。例如：

```
\begin{frame}
\frametitle{Beamer Package}
\begin{itemize}
\item<1-> Normal LaTeX class.
\item<2-> Easy overlays.
\item<3-> No external programs needed.
\end{itemize}
\end{frame}
```

```
\end{frame}
```

其中的 <1->、<2-> 和 <3-> 分别表示在第 1 步、第 2 步和第 3 步开始显示。你可以尝试将其中一项改为 <2-3>，或者将其中两项的参数调换，看看效果如何。

Beamer 中还有更多逐步显示命令,例如 `\uncover`、`\visible`、`\invisible`、`\onslide`、`\onslide+`、`\onslide*` 等,它们的区别如下:

`\uncover<1-2>{text}`: 不显示的内容仍占用空间,可用 `\setbeamercovered` 让它用半透明样式显示

`\visible<1-2>{text}`: 不显示的内容仍占用空间,但是不可能用半透明方式显示出来

`\only<1-2>{text}<1-2>`: 不显示的内容不占用空间,覆盖指令也可以放在后面

`\onslide<1-2>{text}`:可省略 {text},不省略时等同 `\uncover<1-2>{text}`

`\onslide+<1-2>{text}`:可省略 {text},不省略时等同 `\visible<1-2>{text}`

`\onslide*<1-2>{text}`:不可省略 {text},不省略时等同 `\only<1-2>{text}`

## 5.3 主题模板

Beamer 幻灯片的功能很强大,模块化也做得非常好,而且内容和格式也是实现了分离的,可以分别处理。要改变 Beamer 文稿的演示主题,可以用 `\usetheme{主题名}`,其中主题名有如下这些选择:

无导航栏: `default`、`boxes`、`Bergen`、`Pittsburgh` 和 `Rochester`。

带顶栏: `Antibes`、`Darmstadt`、`Frankfurt`、`JuanLesPins`、`Montpellier` 和 `Singapore`。

带底栏: `Boadilla` 和 `Madrid`。

带顶栏和底栏: `AnnArbor`、`Berlin`、`CambridgeUS`、`Copenhagen`、`Dresden`、`Ilmenau`、`Luebeck`、`Malmoe`、`Szeged` 和 `Warsaw`。

带侧栏: `Berkeley`、`Goettingen`、`Hannover`、`Marburg` 和 `PaloAlto`。

这些演示主题可以一个个尝试挑个自己喜欢的来用。另外，这个页面<sup>3</sup>列出了大部分的 Beamer 演示主题的截图，可以直观地对比和选择。

Beamer 的每个演示主题实际上都是由外部主题、内部主题、颜色主题和字体主题这四种细分主题组合而成的。如果要对演示主题作更加细致地选择，可以按照下面这四种细分主题自由组合：

**外部主题** 设定演示文稿是否有顶栏、底栏和侧栏，以及它们的结构，可以用 `\useoutertheme{主题名}` 来选择，其中主题名有如下这些选择：`default`、`infolines`、`miniframes`、`sidebar`、`smoothbars`、`split`、`shadow`、`tree` 和 `smoothtree`。

**内部主题** 设定演示文稿正文内容（例如标题、列表、定理等）的样式，可以用 `\useinnertheme{主题名}` 来选择，其中主题名有如下这些选择：`default`、`circles`、`rectangles` 和 `rounded`。

**颜色主题** 设定演示文稿的各元素配色，可以用 `\usecolortheme{主题名}` 来选择，其中主题名有如下这些选择：`default`、`albatross`、`beaver`、`beetle`、`crane`、`dolphin`、`dove`、`fly`、`lily`、`orchid`、`rose`、`seagull`、`seahorse`、`sidebartab`、`structure`、`whale` 和 `wolverine`。

**字体主题** 设定演示文稿的字体，可以用 `\usefonttheme{主题名}` 命令来选择，其中主题名有如下这些选择：`default`、`serif`、`structurebold`、`structureitalicserif` 和 `structuresmallcapserif`。

Beamer 幻灯片自带的各种模板的配色基本都是惨不忍睹，比如和 pdfscreen 幻灯片的默认配色一比较就差了许多。不过配色可以自己定制，虽然麻烦了点，至少也是可以满足要求的。

## 5.4 中文演示

一般推荐用 `ctex` 宏包处理中文 beamer 文稿，并用 `xelatex` 程序编译，这种情形的例子如下：

```
\documentclass{beamer}
\usepackage[UTF8]{ctex}
```

<sup>3</sup><http://www.hartwork.org/beamer-theme-matrix/>

```
\begin{document}  
\begin{frame}  
中文内容  
\end{frame}  
\end{document}
```

另外，在载入 `ctex` 宏包时加上 `noindent` 选项，可以取消段落的首行缩进。

如果要用传统的 CJK 宏包处理中文 beamer 文稿，并用 `pdflatex` 程序编译，可以用下面的例子：

```
\documentclass[cjk]{beamer}  
\usepackage{CJK}  
%否则编译时遇到中文的章节名会报错  
\hypersetup{CJKbookmarks=true}  
\begin{document}  
\begin{CJK*}{GBK}{song}  
\begin{frame}  
中文内容  
\end{frame}  
\end{CJK*}  
\end{document}
```

## 第六章 高级应用

### 6.1 定义命令和环境

L<sup>A</sup>T<sub>E</sub>X 提供了 `\newcommand` 命令，用于将常用代码块定义为一个新命令。  
比如：

```
\newcommand{\mylim}{\lim\limits_{n\to\infty}}  
$\mylim (x_n+y_n) = \mylim x_n + \mylim y_n$
```

$$\lim_{n \rightarrow \infty} (x_n + y_n) = \lim_{n \rightarrow \infty} x_n + \lim_{n \rightarrow \infty} y_n$$

`\newcommand` 定义的新命令也可以包含参数，比如

```
\newcommand{\myblue}[1]{\textcolor{blue}{#1}}  
\myblue{Some Text}
```

Some Text

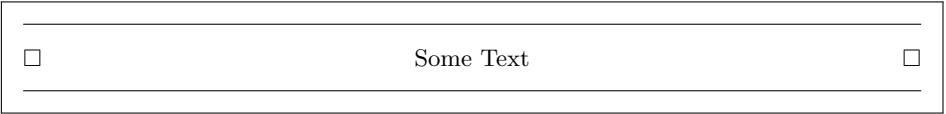
其中可选参数里面的 1 表示新命令的参数个数，最多可以有 9 个。这些参数在定义时分别用 #1 到 #9 表示。

还有一个类似的 `\newcommand*` 命令。这个命令与 `\newcommand` 命令的区别是，前者所定义的命令不能包含多个段落（比如里面不能出现空行或者 `\par` 命令）。

利用 `\newenvironment` 命令可以定义一个新环境。比如



```
% define a new environment
\newenvironment{myenv}
  {\hrule\vspace{1em}$\square$\hfill}
  {\hfill$\square$\vspace{1em}\hrule}
% use this new environment
\begin{myenv}Some Text\end{myenv}
```



同样地,用 `\newenvironment` 定义的新环境也可以有参数。比如

```
% define a new environment
\newenvironment{mycolorenv}[1]
  {\color{#1}$\square${$\square$}}
% use this new environment
\begin{mycolorenv}{blue}
  Some Text
\end{mycolorenv}
```



## 6.2 使用计数器

$\text{\LaTeX}$  的自动编号依赖于各种计数器。比如节编号依赖于 `section` 计数器,我们用 `\arabic{section}` 显示当前的节计数器的数值:

当前段落位于第 `\arabic{section}` 节。

当前段落位于第 2 节。

实际上,计数器的数值有多种显示方式:

<code>\arabic</code>	<code>\alph</code>	<code>\Alph</code>	<code>\roman</code>	<code>\Roman</code>
1, 2, 3, ...	a, b, c, ...	A, B, C, ...	i, ii, iii, ...	I, II, III, ...

计数器也可以用于整数比较,此时要用 `\value` 命令。比如

```
\ifnum \value{section}=2 Yes \else No \fi
```

Yes

在节标题中,节编号的显示方式记录在 `\thesection` 命令中。比如要得到与本书一样的节编号样式,可以用如下命令

```
\renewcommand{\thesection}{\arabic{chapter}.\arabic{section}}
```

L<sup>A</sup>T<sub>E</sub>X 预先定义了多个计数器，常见的有 `page` 这个页码计数器；`part`，`chapter`，`section`，`subsection` 等章节计数器；`figure` 和 `table` 这两个图表计数器；以及 `enumi`，`enumii`，`enumiii`，`enumiv` 这四个有序列表计数器。因为有序列表可以嵌套，所以需要多个计数器。

利用计数器，我们也可以自己定义带编号的环境，比如：

```
% create a new counter
\newcounter{mycounter}
% define a new environment with this counter
\newenvironment{mytheorem}
  {\par\stepcounter{mycounter}Theorem-\arabic{mycounter}\quad}
  {\quad$\square$}
% use this new environment twice
\begin{mytheorem}
  Some Text.
\end{mytheorem}
\begin{mytheorem}
  More Text.
\end{mytheorem}
```

Theorem 1	Some Text.	<input type="checkbox"/>
Theorem 2	More Text.	<input type="checkbox"/>

在这个例子中，我们用 `\newcounter` 命令定义一个新计数器，用 `\stepcounter` 命令给计数器的数值加一。

如果要让计数器编号可以被 `\ref` 命令引用，只需要将 `\stepcounter` 改为 `\refstepcounter`。

另外还有 `\addtocounter` 命令，它用于给计数器增加任意的数值；以及 `\setcounter` 命令，它用于直接设定计数器的数值。比如

```
\addtocounter{mycounter}{3}
\setcounter{mycounter}{8}
```

## 6.3 处理错误和警告

对于正确的 tex 文档，例如我们的第一个文档 `hello.tex`，编译时显示的信息类似于下面的例子：

```
This is XeTeX, Version 3.1415926-2.2-0.9997.4 (MiKTeX 2.9)
entering extended mode
(E:\work\latex\hello.tex
LaTeX2e <2009/09/24>
(D:\CTEX\MiKTeX\tex\latex\base\article.cls
(D:\CTEX\MiKTeX\tex\latex\base\size10.clo))
No file hello.aux.
[1] (E:\work\latex\hello.aux) )
Output written on hello.pdf (1 page).
SyncTeX written on hello.synctex
Transcript written on hello.log.
```

这些信息告诉我们，`hello.tex` 文档已经编译成功，同时编译的信息已经写在 `hello.log` 文件中了。

### 6.3.1 编译时的错误信息

如果编译时遇到错误，编译就会暂停，并且显示出错信息等待我们处理。最常见的错误信息如下：

```
! Undefined control sequence.
1.3 \hello
      world
?
```

这个信息表示编译程序遇到了未定义的命令 `\hello`，同时指明这个错误出现在第 3 行。在 `?` 号后面你可以直接按回车键让它继续编译，或者按 `r` 键让它继续编译而且不再显示错误。

对于这种最简单的错误，基本的处理方式如下：首先检查是否这个命令的某个字母写错了；其次检查是否这个命令所依赖的宏包未在导言区载入；如果你确认代码没有问题，有可能是辅助文件的问题，你可以删除 `tex` 文档所在目录的 `.aux`、`.idx`、`.ind`、`.out`、`.toc` 等等文件后重新编译试试。

### 6.3.2 编译时的警告信息

编译时还可能碰到各种警告信息，常见的警告信息类似于下面这些情形：

```
Overfull \hbox (8.78062pt too wide) in paragraph at lines 427--427  
Overfull \vbox (6.56552pt too high) has occurred while \output is active []  
Underfull \hbox (badness 10000) in paragraph at lines 488--498
```

上面例子的第一行是行溢出警告，即 TeX 程序不能很好的断行，导致超出边界 8.78062pt 的长度。这个警告通常在某行有多个太长的单词时出现。

第二行是页溢出警告，即 TeX 程序不能很好的分页，导致文本延伸到页面底部。这个警告通常在文档中有大型表格或者图片时出现。

第三行是行稀疏警告，即 TeX 程序在某些行不能放入足够多的单词，导致单词之间空隙太大有碍观瞻。这个警告在手工断行时会出现。其中的 **badness** 值越大表示情况越糟糕，而且超过 10000 后都显示 10000。

# 第七章 字体详解

## 7.1 字体类型与文档格式

大家都说 Word 容易入门很难深入，而 TeX 的入门和深入都不容易。TeX/LaTeX 的复杂有一个原因在于它的名目繁多的字体类型和文档格式。这里将总结下这两方面的问题<sup>1 2</sup>。

首先是字体类型。LaTeX 中遇到的字体类型至少有 pk, metafont, type1, truetype 和 opentype 等等。pk 字体是点阵字体（在 pdf 文件中被视为 type3 字体），它可由 metafont 和 truetype 等矢量字体生成。type1 和 opentype 也是矢量字体，这四种矢量字体依出现的先后依次为 metafont, type1, truetype 和 opentype。（严格说来每个 metafont 和 type1 字体都由两个文件组成，这里不详叙。）

mktexpk 程序可从 metafont 字体生成 pk 字体，ttf2pk 程序可以从 ttf 字体生成 pk 字体，而 ttf2pfb 和 ttf2pt1 程序可以从 ttf 字体生成 type1 字体。

其次是文档格式。LaTeX 中遇到的文档格式至少有 tex, dvi, ps 和 pdf。用 latex 程序可将 tex 文档编译为 dvi 文档，然后用 dvips 程序可以将 dvi 文档转换为 ps 文档，最后用 ps2pdf 程序可以将 ps 文档转换为 pdf 文档。

接着出现的 dvipdf 程序其实是一个小脚本，它将 dvips 和 ps2pdf 这两个程序连接起来，将 dvi 文档间接地转换到 pdf 文档。后来出来个 dvipdfm 程序，才真正地做到将 dvi 文档直接转换为 pdf 文档。然后在其基础上又出来个 dvipdfmx 程序，加强了对中日韩字体的支持。还有个 pdflatex 程序，可以直接将 tex 文档编译为 pdf 文档，最是简单快捷。

为使生成的 pdf 文档更加美观，特别是放大之后不产生锯齿，就必须在 pdf 文档中使用矢量字体。dvips 和 dvipdfm 只能使用 pk 点阵字体和 type1

---

<sup>1</sup>吴凌云, CTeX 常用问题集 v0.4 beta, 2005.

<sup>2</sup>jackqq, LaTeX 中文排版上手随记, 2008.

矢量字体，而 dvipdfmx 和 pdflatex 可以使用 pk 点阵字体，type1 和 truetype 矢量字体。因此，为了得到高质量的 pdf 文档，最好用 dvipdfmx 和 pdflatex 来生成文档。

最新的 xelatex 程序也是将 tex 文档直接编译为 pdf 文档，而且它可以使用前面所说的所有字体，独特的是只有它能支持 OpenType 字体的高级特性。

## 7.2 新字体选择方案 NFSS

在高德纳的 Plain TeX 中，使用新字体前需要先知道字体的文件名，接着用 `\font` 命令定义新字体，然后才能使用这个字体。例如：

```
\font\newfont=cmr10 at 20pt
{\newfont hello font!}
```

hello font!

这个例子中，第一行将 cmr10 字体缩放到 20pt 大小定义为 newfont 字体。然后第二行用这个新字体显示 hello font! 文本。其中的 cmr10 指的是 TeX 安装目录里的 fonts/source/public/cm/cmr10.mf 这个 Metafont 字体文件。

后来，随着技术的发展，各种新字体层出不穷，这种原始的使用字体方式就有些落后了。于是，在 LaTeX 的 2 $\epsilon$  版本中正式引入了新字体选择方案 (NFSS)。NFSS 是一种方便地选择和使用字体的方案<sup>3</sup>。NFSS 把一个字体的属性分为 5 个：编码、族名、系列、形状和大小。LaTeX 中已经预先定义了几种编码：比如 OT1 和 OML、OMS、OMX 分别是高德纳原来的计算机现代文本字体和数学字体的编码。这些字体编码可以直接使用。你可以在 TeX 系统的 tex\latex\base 目录里找到 ot1enc.def、omsenc.def 等字体编码定义文件。

要使用其它的字体，首先用下面的命令设置一个新的编码：

```
\DeclareFontEncoding{编码}{文本模式命令}{数学模式命令}
```

比如 CJK 中就定义了各种中日韩文的编码：比如 C00 是繁体中文的 Bg5 编码，C10 和 C19 分别是简体中文的 GB 和 GBK 编码，C70 是 Unicode 的 UTF8 编码，等等。这些定义在 CJK.enc 文件中。

接着定义某种编码的字体属性可以用如下命令：

```
\DeclareFontFamily{编码}{族名}{命令序列}
\DeclareFontShape{编码}{族名}{系列}{形状}{字体定义}{命令序列}
```

<sup>3</sup>Ulrik Vieth & Taco Hoekwater, Surviving the TEX font encoding mess, 1999.

这些定义在若干 CJK 目录里的若干.fd 文件中：比如 `c10gbsn.fd`、`c19song.fd` 等等，其中文件名开头的 `c10` 和 `c19` 等指的是上面定义的编码 (fontencoding)，而后面的 `gbsn` 和 `song` 指的是字体的族名 (fontfamily)。

用这个新字体选择方案来添加并使用新字体，仍然不是个方便的方法。现在建议使用 XeTeX 来排版中文 LaTeX 文档。

## 7.3 中文排版历程

### 7.3.1 CCT 中文排版

使用 CCT 排版中文 LaTeX 文档是不推荐的做法，这里简要介绍，以供参考。我们先看例子：

```
\documentclass{cctart}
\begin{document}
\songti 宋体是CCT的默认字体。
\heiti 现在是黑体。
\kaishu 现在是楷书。
\fangsong 现在是仿宋。
\end{document}
```

这里的 `cctart` 根据实际情况可以换为 `cctbook`，分别与英文 latex 的 `article` 和 `book` 文档类别对应。

CCT 有先后两种版本，旧版 CCT 文档的文件名是以 `.ctx` 为扩展名的，而新版 CCT 文档的文件名以 `.tex` 为扩展名，推荐使用新版的 CCT，因为新版的 CCT 相比旧版的 CCT 有不少优点，比如它可以使用 CJK 宏包，只要在开始加入 CJK 参数就可以了：

```
\documentclass[CJK]{cctart}
\begin{document}
你好！这里是CCT！
\end{document}
```

CCT 文档可以用 `ctex.exe` 程序编译就可以了（对于 CTeX 套装，点击 WinEdt 里的 CCT-LaTeX 图标）。`ctex.exe` 可以自动识别是 `ctx` 格式还是 `tex` 格式，然后自动调用相应的程序去处理 `cct` 文档。

注记：CCT 中没有定义隶书和幼圆这两种字体。

注意：使用 CCT 来排版中文 LaTeX 文档不太方便，现在建议使用 XeTeX 来排版中文 LaTeX 文档。

### 7.3.2 CJK 中文排版

使用 LaTeX 排版中文文档，传统的方法有两种：一是用 CJK 宏包，二是用 CCT 宏包。这里先介绍 CJK 的中文排版。

CJK 是外国人编写的，不仅支持中文，而且也支持其它双字节语言（实际上 CJK 就是 Chinese, Japanese 和 Korean 这三个词语的缩写）。MiKTeX 和 CTeX 中都有这个包，安装之后就可以这样使用了：

```
\documentclass{article}
\usepackage{CJK}
\begin{document}
\begin{CJK}{GB}{gbsn}
欢迎来到CJK！这里使用的是CJK宏包里有的文鼎宋体。
\end{CJK}
\end{document}
```

然后就用 latex 编译就可以得到中文文档了，不比英文排版复杂多少，关键是要把中文部分放在 `\begin{CJK}` 和 `\end{CJK}` 之间。上面代码中的 GB 指的是编码，对于简体中文来说，常见的有 GB 编码和 GBK 编码两种。而 gbsn 指使用的字体为文鼎宋体。

由于字体的版权问题，CJK 的宏包里只有文鼎宋体和文鼎楷体这两种字体，可是这两种字体生成的中文文档的效果很一般。CTeX 里另外多了 GBK 编码的六种中文字体（宋体、仿宋、楷体、黑体、隶书和幼圆），如果你安装了 CTeX，就可以类似下面的例子来使用这几种字体：



```
\documentclass{article}
\usepackage{CJK}
%\begin{document}
\begin{CJK}{GBK}{song}

CTeX里中文默认用宋体！
\CJKfamily{GBK}{hei} 这是CTeX里的黑体！
\CJKfamily{fs} 这是CTeX里的仿宋体！
\CJKfamily{kai} 这是CTeX里的楷体！
\CJKfamily{li} 这是CTeX里的隶书！
\CJKfamily{you} 这是CTeX里的幼圆体！

\end{CJK}
\end{document}
```

注记：中文 Windows XP 系统里没有隶书和幼圆这两种字体，而英文 Windows XP 系统里另外还没有楷体和仿宋这两种字体。这几个字体在安装微软 Office 时会安装。如果只安装 CTeX 而没安装 CTeX-Fonts 的话，编译 CJK 文件时需要用到系统自带的中文 ttf 字体文件，这样就会出错。

使用 CJK 来排版中文 LaTeX 文档不太方便，现在建议使用 XeTeX 来排版中文 LaTeX 文档。

### 7.3.3 XeTeX 中文排版

之前已经介绍过用 CCT 和 CJK 宏包来排版中文 LaTeX 文档。用这两者来排版中文文档都是权宜之计，因为原本的 TeX 排版程序就不支持中文，CCT 和 CJK 就是两种外挂而已，很容易就出现比如 pdf 书签乱码之类的问题。新近的 XeTeX 和 LuaTeX 从底端就支持各种文字包括中文，而且能够直接使用系统自带的字体，生成的文档更为美观，所以从 TeX 转换到 XeTeX/LuaTeX 来自然是潮流之所趋。LuaTeX 目前基本不支持 LaTeX 文档，所以现在先介绍用 XeTeX 排版 LaTeX 文档的方法<sup>4 5 6 7 8</sup>。

实际上，XeTeX 对 TeX 的改动不大，主要集中在字体的使用上，因此原来

---

<sup>4</sup>lyanry@gmail.com, XeTeX 中文排版之胡言乱语, 2007.

<sup>5</sup>kmc@bbs.ctex.org, XeTeX about:fonts, 2008.

<sup>6</sup>kmc@bbs.ctex.org, LaTeX 发行版自带的字体, 2008.

<sup>7</sup>Marko Boon, XeTeX in MiKTeX, 2008.

<sup>8</sup>Michel Goossens, The XeTeX Companion: TeX meets Opentype and Unicode, 2010

的 LaTeX 文档稍为修改下就能够用 XeTeX 编译了。最新的 MiKTeX (CTeX) 和 TeXLive 套装都包含 XeTeX 程序, 编辑器推荐用免费的 TeXworks。在编辑器中输入如下文本

```
% !TEX program = xelatex

\documentclass[12pt,a4paper]{article}

\usepackage[cm-default]{fontspec}
\usepackage{xunicode}
\usepackage{xltextra}

\setmainfont[BoldFont=SimHei,ItalicFont=KaiTi_GB2312]{SimSun}
\setsansfont[BoldFont=SimHei]{KaiTi_GB2312}
\setmonofont{NSimSun}

\XeTeXlinebreaklocale "zh"
\XeTeXlinebreakskip = 0pt plus 1pt

\begin{document}

\XeTeX\ Show: $\alpha$, $a^2+b^2=c^2$

中文字体!

\end{document}
```

然后使用 xelatex 编译, 就得到一个中文文档了。一言以蔽之, 很好很强大。

注意 WinEdt 对 UTF-8 编码支持不够完善, 需要手动选择; 或者在新建的 tex 文件的首行写上如下这行

```
% !Mode:: "TeX:UTF-8"
```

以让 WinEdt 在打开和保存时作编码转换。即使这样, 有时也会造成乱码或字符丢失, 所以不推荐用 WinEdt 编辑 UTF-8 编码的 xetex+latex 文档。

## 附录 A 常用宏包介绍

`alltt` 特殊的抄录环境，命令保持有效（即 `\ { }` 这三个字符意义不变）。例如

```
\begin{alltt}
\textit{// this is some comments.}
void main \{
    printf ("Hello World!");
\}
\end{alltt}

// this is some comments.
void main {
    printf ("Hello World!");
}
```

网址: <http://www.ctan.org/pkg/alltt>。

`amscd` 画矩形交换图，参考 3.6.1。

网址: <http://www.ctan.org/pkg/amscd>。

`amsmath` 数学公式增强版，参考 3.5。

网址: <http://www.ctan.org/pkg/amsmath>。

`amssymb` 使用黑板体和花体等数学字体，参考 3.2。

网址: <http://www.ctan.org/pkg/amsmath>。

`beamerarticle` 在 `article` 文档类中使用 `beamer` 命令，参考第 5 章。例如：

```

\documentclass{article}
\usepackage{beamerarticle}
\begin{document}
Hello \pause World!
\end{document}

```

网址: <http://www.ctan.org/pkg/beamer>。

CJK 使用中日韩语言。

网址: <http://www.ctan.org/pkg/cjk>。

color 使用背景和前景颜色。

网址: <http://www.ctan.org/pkg/color>。

comment 增强的注释环境, 而且可以添加新注释环境, 并设定是否包含。例如:

```

\includecommnt{newcmta}
\excludecommnt{newcmtb}
\begin{newcmta}
These comments are included.
\end{newcmta}
\begin{newcmtb}
These comments are excluded.
\end{newcmtb}

```

网址: <http://www.ctan.org/pkg/comment>。

ctex 设置中文环境。

网址: <http://www.ctan.org/pkg/ctex>。

ctexcap 设置中文环境和标题。

网址: <http://www.ctan.org/pkg/ctex>。

diagrams 画交换图。例如:

```

\begin{diagram}
L_K & \rTo & M_K \\
& \rdTo \rdTo(2,4) & \dTo \\

```

```

& &W_K \\
& &\dTo \\
& &G_K \\
\end{diagram}

```

网址: <http://www.ctan.org/pkg/taylor>。

`docmute` 忽略所包含文件的导言区。利用这个宏包, 我们可以将大型文档拆分成多个部分, 而且保证各部分仍然可以独立编译。例如:

```

% main file
\documentclass{article}
\usepackage{docmute}
\input{preamble}
\begin{document}
\input{sub1}
\input{sub2}
\end{document}

% sub file
\documentclass{article}
\input{preamble}
\begin{document}
\section{Section One}
Some Text.
\end{document}

```

其中 `preamble` 文件包含在导言区中所作的各种定制。

网址: <http://www.ctan.org/pkg/docmute>。

`fancyvrb` 提供在脚注中使用行内抄录命令 (`\VerbatimFootnotes`), 自定义分隔符的行内抄录命令 (`\Verb`), 以及高级的抄录环境 `Verbatim`。下面的例子取自其文档:

```

\VerbatimFootnotes
We can put verbatim\footnote{\verb+_Yes!_+} text in footnotes.

```

```
\DefineShortVerb{\|}  
We can simply write \Verb+_verbatim_  
material using a single |_delimiter_|  
\UndefineShortVerb{\|}  
\DefineShortVerb{\+}  
And we can +_change_+ the character.
```

```
\begin{Verbatim}  
First verbatim line.  
Second verbatim line.  
\end{Verbatim}
```

网址: <http://www.ctan.org/pkg/fancyvrb>。

`float` 增加浮动对象。例如:

```
\newfloat{program}{htbp}{lop}[section]  
\begin{program}  
...  
\end{program}
```

网址: <http://www.ctan.org/pkg/float>。

`fontspec` 设置 XeTeX 字体。

网址: <http://www.ctan.org/pkg/fontspec>。

`geometry` 调整页面布局。

网址: <http://www.ctan.org/pkg/geometry>。

`graphics` 插入图片。

网址: <http://www.ctan.org/pkg/graphics>。

`graphicx` 插入图片, 比 `graphics` 宏包高级。

网址: <http://www.ctan.org/pkg/graphicx>。

`hyperref` 在生成的 PDF 文档中添加链接以及书签和属性。例如:

```

\hypersetup{
  pdftitle={LaTeX Log},
  pdfauthor={zoho@bbs.ctex.org},
  colorlinks,
  citecolor=blue,
  filecolor=blue,
  linkcolor=blue,
  urlcolor=black
}
\url{http://www.ctex.org}
\href{http://www.ctex.org}{CTeX Homepage}

```

网址: <http://www.ctan.org/pkg/hyperref>。

`ifthen` 逻辑判断语句。例如:

```

\ifthenelse{\isodd{\value{page}}}{
  {hello odd page}{hello even page}
}

```

网址: <http://www.ctan.org/pkg/ifthen>。

`indentfirst` 对章节的第一段同样缩进。

网址: <http://www.ctan.org/pkg/indentfirst>。

`index` 编写多个索引。

网址: <http://www.ctan.org/pkg/index>。

`lastpage` 记录末尾页的页码, 可以用命令 `\pageref{LastPage}` 调用。

网址: <http://www.ctan.org/pkg/lastpage>。

`layout` 用 `\layout` 命令生成当前页面的示意图。

网址: <http://www.ctan.org/pkg/layout>。

`listings` 提供可作为命令参数的行内抄录命令 `\lstinline`, 以及可以指定代码语言, 背景颜色等属性的行间抄录环境 `lstlisting`。

网址: <http://www.ctan.org/pkg/listings>。例如:

```

\lstset{

```

```

basicstyle=\ttfamily\small,
backgroundcolor=\color{lightgray},
escapechar=`,
xleftmargin=8pt,
framexleftmargin=8pt,
}
\begin{lstlisting}
...
\end{lstlisting}

```

`longtable` 跨页的长表格。

网址: <http://www.ctan.org/pkg/longtable>。

`makeidx` 生成名词索引。

网址: <http://www.ctan.org/pkg/makeidx>。

`mdwlist` 提供紧凑的列表环境 `itemize*`、`enumerate*` 和 `discription*`。可以比较下面例子的结果:

```

\begin{enumerate}
\item First item
\item Second item
\end{enumerate}
\begin{enumerate*}
\item First item
\item Second item
\end{enumerate*}

```

1. first item
2. second item
3. third item

1. first item
2. second item
3. third item

网址: <http://www.ctan.org/pkg/mdwlist>。



metalogo 提供各种 TeX 标识。

网址: <http://www.ctan.org/pkg/metalogo>。

multicol 提供 multicol 环境用于多栏排版。

网址: <http://www.ctan.org/pkg/multicol>。

multidx 编写多个索引。

网址: <http://www.ctan.org/pkg/multidx>。

multirow 编写跨行表格。

网址: <http://www.ctan.org/pkg/multirow>。

paralist 提供紧凑的列表环境 (compactitem, compactenum 和 compactdesc) 以及行内的列表环境 (inparaitem, inparaenum 和 inparadesc)。例如:

```
\begin{inparaitem}
\item first item
\item second item
\item third item
\end{inparaitem}
```

• first item • second item • third item

网址: <http://www.ctan.org/pkg/paralist>。

pdfpages 包含 PDF 文档。从有些地方 (比如施普林格网站) 下载的电子书籍是按照每章一个 PDF 文件的方式分开的, 这样不方便阅读和管理。如果要将一些 PDF 文件合并为一个, 可以用这个宏包。例如:

```
\documentclass{article}
\usepackage{pdfpages}
\begin{document}
\includepdfmerge{Lang1.pdf,-}
\includepdfmerge{Lang2.pdf,-}
\includepdfmerge{Lang3.pdf,-}
\end{document}
```

注意编译时要用 pdflatex 程序, 而不能用 latex 程序。

网址: <http://www.ctan.org/pkg/pdfpages>。

pdfslide 制作演示文稿。

网址: <http://www.ctan.org/pkg/pdfslide>。

pdfscreen 制作演示文稿。

网址: <http://www.ctan.org/pkg/pdfscreen>。

realscripts 在文本环境中使用上下标。

网址: <http://www.ctan.org/pkg/realscripts>。

shortverb 行内的抄录环境, 对 \verb 命令的改进。

网址: <http://www.ctan.org/pkg/shortverb>。

showidx 将索引项显示为边注。

网址: <http://www.ctan.org/pkg/showidx>。

splitidx 编写多个索引, 个数不限制; 但需要使用另外的处理程序。

网址: <http://www.ctan.org/pkg/splitidx>。

tabularx 指定宽度的表格。例如

```
\begin{tabularx}{0.9\textwidth}{|c|X|X|}  
\hline  
Small width & Large width & Large width \\  
\hline  
Small width & Large width & Large width \\  
\hline  
\end{tabularx}
```

网址: <http://www.ctan.org/pkg/tabularx>。

texpower 演示文稿的动画效果。

网址: <http://www.ctan.org/pkg/texpower>。

tikz 绘制 PGF 矢量图形。

网址: <http://www.ctan.org/pkg/pgf>。

titlesec 设置标题格式。

网址: <http://www.ctan.org/pkg/titlesec>。

`titletoc` 设置目录页格式。

网址: <http://www.ctan.org/pkg/titletoc>。

`verbatim` 增强的抄录环境, 以及基本的注释环境。

网址: <http://www.ctan.org/pkg/verbatim>。

`wasysym` 使用各种特别符号。例如:

`\davidssstar ☆ \leftmoon ☾ \rightmoon ☽ \sun ☼`

网址: <http://www.ctan.org/pkg/wasysym>。

`xcolor` 使用颜色, 比 `color` 宏包高级。

网址: <http://www.ctan.org/pkg/xcolor>。

`xltxtra` 用于 XeTeX 文档, 载入 `fontspec`、`metalogo` 和 `realscripts` 宏包, 并添加一些命令。

网址: <http://www.ctan.org/tex-archive/macros/xetex/latex/xltxtra>。

`xunicode` 用于 XeTeX 文档, 处理 Unicode 字符。

网址: <http://www.ctan.org/pkg/xunicode>。

`xy` 绘制各种图表, 包括交换图和示意图等。

网址: <http://www.ctan.org/pkg/xypic>。

# 附录 B TeX 背景资料

## B.1 TeX 系统的前世今生

大家都知道, TeX 是大牛 Knuth (高德纳) 在 70 年代末设计的一个排版系统, 它包括 TeX 排版程序和 Plain TeX 宏集这两部分。Plain TeX 可以看成是一种编程语言, 我们按照它的规则来写一个以 .tex 为后缀的文件, 然后用 TeX 排版程序编译就得到 dvi 文档。dvi 文档可以转换成 ps 或 pdf 文档, 用于打印或出版。TeX 排版程序有许多改进版, 当前最流行的是 pdfTeX, 它包含在 MiKTeX 和 TeXLive 等 TeX 套装里面。pdfTeX 最大的优点是可以将 tex 文件直接编译为 pdf 文档。

但是 Plain TeX 宏集实在是太低级了, 不容易掌握。于是就有了各种建立在 Plain TeX 之上的更高级的宏集, 大大降低了排版的难度。在这些高级的宏集中以 80 年代设计的 LaTeX 宏集和 90 年代设计的 ConTeXt 宏集较为出名。如果把 Plain TeX 宏集看成汇编语言的话, LaTeX 宏集和 ConTeXt 宏集就类似于 C 或者 Pascal 编程语言。用 LaTeX 或者 ConTeXt 写的 tex 文档分别需要用 latex 和 context 命令编译, 这两个命令其实都调用了某个 TeX 排版程序 (比如 pdfTeX)。

原始的 TeX 排版程序在现在看来有两个问题, 就是使用新字体 (比如 TrueType 字体和 OpenType 字体) 麻烦, 对中日韩等各种语言的支持也很麻烦。虽然你用 CCT 或者 CJK 写中文文档看来还好, 实际上内部的实现要多丑陋有多丑陋。于是到了 21 世纪就有了 XeTeX 和 LuaTeX 这两个原始 TeX 排版程序的替代者出来了。XeTeX 排版程序的版本号已经到了 0.99, 应该快要正式发布了; 经过测试, 中文排版基本没问题, 排版的效果也好多了。LuaTeX 才到 0.70 版本, 短期内还未能正式发布, 它的最大的特点就是引入了一种轻巧的脚本语言 lua, 这使得它的可扩展性相当的大。XeTeX 基本兼容 Plain TeX, LaTeX 和 ConTeXt 宏集, 这些文档分别用 xetex, xelatex 和 xecontext 命令

来编译。就目前的发展看来, XeTeX+LaTeX 和 LuaTeX+ConTeXt 是比较有前途的两个发展方向, 前者稍为简单和传统, 后者偏于灵活和新颖。

最后说说关于 TeX 系统未来的个人看法。从现在的趋势看, 首先 dvi, ps 文件是会慢慢消失的, 因为现在 pdftex 程序可以直接从 tex 文档生成 pdf 文档, 而 synctex 已经可以实现 tex 文档和 pdf 文档的互相跳转。其次 CCT, CJK 等宏包也是会慢慢消失的, 因为用 xetex 已经能够方便地进行中文排版了。最后, pk 字体 (Type 3 字体) 也是会慢慢消失的, 因为 TrueType 字体和 OpenType 字体显示的效果比它们美观多了。

## B.2 TeX 系统的目录结构

TeX 系统目录结构 (TeX Directory Structure) 简称 TDS, 是 TUG (TeX Users Group) 主持制定的标准, 目的在于方便 TeX 的开发者和用户。目前流行的 MiKTeX 套装和 TexLive 套装都支持 TDS。

TDS 由几个结构类似的树状目录组成, 主要有 TEXMF 和 LOCAL-TEXMF 这两个树状目录, TDS 并未规定这两个目录的名称, 而是规定各种 TeX 文件在这两个目录里的存放位置。其中 TEXMF 目录一般存放 TeX 发行版的文件, LOCALTEXMF 目录一般存放本地添加的各种 TeX 文件。

如果你安装了基于 MiKTeX 的 CTeX 套装, CTeX 目录里的 texmf 和 localtexmf 目录就是上述的主要目录了; 在 texmf 目录里存放的是 MiKTeX 发行版的各种 TeX 文件, 而在 localtexmf 目录里存放的是 CTeX 套装添加的各种中文相关的文件, 比如 CCT 和一些中文教程等等。下面是 texmf 目录里的各种子目录及其解释:

1. bibtex, dvipdfm, dvips, etex, metafont, metapost, omega, pdftex 这些目录里分别存放的是各个程序的配置文件等。其中 dvips 和 dvipdfm 分别是将 dvi 文档转化为 ps 和 pdf 文档的程序; metafont 是 TeX 自带的字体生成程序, metapost 是从 metafont 发展而来的矢量画图程序; 而 etex, omega 和 pdftex 是原始 tex 程序的各种变种, 目前以 pdftex 最流行。

2. doc 目录里存放的是各种程序和宏集的说明文档; source 目录里存放各种源代码。

3. tex 目录是核心部分之一, 里面存放各种宏集语言的类文件和宏包, 一种宏集格式为一个子目录, 包括 amstex, context, latex, plaintex, texinfo 等等。在这些子目录里面, 都有一个 base 目录用于存放该宏集语言的基本文件, 适用于该语言的各种宏包各自放在自己的目录里。比如你下载了一个 latex 宏

包，名字为 `ltxpkg`，那么你应该在 `texmf\tex\latex` 目录里新建 `ltxpkg` 子目录并将该宏包的所有文件放入其中。`tex` 目录里面还有个 `generic` 子目录，该目录存放适用于多种宏集语言的宏包，比如某些画图的宏包等。

3. `fonts` 目录是核心部分之二，里面存放各种格式的字体文件和字符映射文件，一般也是每种字体格式占用一个子目录。比如：`tfm` 子目录存放 TeX 的字体度量文件（不包含各字符的形状信息），`source` 子目录存放 `metafont` 字体形状文件，`pk` 子目录存放压缩的点阵字体文件，`vf` 子目录存放虚拟字体文件。`afm`，`pfm` 和 `type1` 子目录分别存放 PostScript Type 1 矢量字体的各种后缀的文件。`truetype` 和 `opentype` 子目录分别存放 TrueType 和 OpenType 这两种矢量字体文件。`ofm`，`ovf` 和 `ovp` 子目录存放的是 `omega` 排版程序的各种字体文件；`omega` 是原始 `tex` 程序的变种，主要目的是支持 Unicode，现在已经被废弃了，XeTeX 和 LuaTeX 是未来的发展方向。

4. `miktex` 目录存放 MiKTeX 套装的各种配置文件以及应用程序（而 `web2c` 目录是 TeXLive 套装的各种配置文件的存放目录）。

注记 1: Type 1 是 Adobe 公司提出的一种矢量字体标准。每个 Type 1 字体由两个文件组成：一个 `pfa` 或 `pfb` 文件和一个 `afm` 或 `pfm` 文件。第一个文件描述各个字符的形状，第二个文件描述各个字符的度量。`pfa` 文件和 `pfb` 文件基本等价，区别在于前者是 `ascii` 文件而后者是二进制文件；`afm` 文件和 `pfm` 文件也基本等价，区别在于前者为 `ascii` 文件，多出现在 Linux 系统，而后者为二进制文件，多出现在 Windows 系统。

注记 2: 最开始的 TeX 使用的 Computer Modern 字体由 `tfm` 文件和 `mf` 文件组成（类似于 Type1 字体的 `afm` 文件加 `pfb` 文件），用 `tex` 程序将 `.tex` 文档编译成 `.dvi` 文档时只需要用到 `tfm` 文件，`dvi` 阅读器（比如 `yap` 程序）在打开 `.dvi` 文件时将调用 `metafont` 程序（`mf.exe`）从 `mf` 文件中生成 `pk` 点阵文件以供显示。要使用 Type1 字体，首先得先将 `pfm` 或 `afm` 文件转换为 `tfm` 文件，`dvi` 阅读器在打开使用这些字体的 `.dvi` 文档时将调用 `ps2pk` 程序从 `pfa` 或 `pfb` 文件中生成 `pk` 点阵文件以供显示。

## B.3 TeX 系统的文件查找

TeX 发展了若干年，各种字体，宏包的数目已经非常地多了，比如你看现在的 CTeX 完整版安装完后就已经超过 1GB 大小了，而 TeXLive 光盘已经要使用 DVD 了。这么多的文件，各个 `tex` 编译程序如何找到它们就成了个大问题，总不能一个目录一个目录地搜索吧。

TeX 用户组织 (TUG) 发明了 Web2C/Kpathsea 系统, 用于自个发布的 TeXLive 发行版。这个系统基本思路是: 所有 tex 相关文件按照 TeX 目录结构标准 (TDS) 放在若干个 TEXMF 目录里面, 然后所有 tex 程序比如 tex.exe, pdftex.exe, xetex.exe 在编译的时候如果需要寻找任何文件, 都通过 kpathsea.dll 运行库查询文件所在的位置。当然, 你也可以自己运行 kpsewhich.exe 程序手动查询某个文件所在的位置。

kpathsea 当然不会傻到一个目录目录的去查找文件, 它是根据 tds 目录结构, 按照文件的扩展名去相应的目录查找。比如你需要寻找 article.cls 文件, 运行下列命令

```
kpsewhich.exe article.cls
```

时, kpathsea 发现这个文件是 latex 的文档类文件, 所以只去各个 TEXMF 目录的 tex\latex 子目录里查找这个文件。然后返回文件所在的位置。

我们之前说 TEXMF 目录有不止一个, 比如你打开 TeXLive 的安装目录, 可以看到 texmf, texmf-dist, texmf-var 等等子目录, 这些都是 TEXMF 目录。所以现在还得明白的是, kpathsea 怎么才能知道 TEXMF 目录都有哪些, 分别在什么位置呢? 实际上 kpathsea 运行的时候, 首先要去寻找一个叫做 texmf.cnf 的文件, 这个文件记录了各个 TEXMF 所在的位置。

那么, kpathsea 又要去哪些地方寻找 texmf.cnf 文件呢? 在 kpathsea 中, 以 \$SELFAUTOLOC、\$SELFAUTODIR 和 \$SELFAUTOPARENT 变量分别表示 kpathsea 所在目录、kpathsea 所在目录的上一级目录和 kpathsea 所在目录的上两级目录。默认情况下, kpathsea 按照如下顺序查找 texmf.cnf 文件:

```
{ $SELFAUTOLOC, $SELFAUTODIR, $SELFAUTOPARENT } {, {/share, }/texmf{-local, }/web2c }
```

这种写法是 kpathsea 的简写, 需要解释一下: 花括号里面用逗号隔开的几个路径段表示要挨个试, 括号与括号或者路径段放在一起表示两者连接起来组成新的路径; 而特殊的是, 括号内的最前或最后可能有一个逗号, 这表示最前面或最后面有一个空路径段。这种简写法多半很雷人, 那么我们把这个表示法的完整展开式按顺序写一下吧:

```
$SELFAUTOLOC;  
$SELFAUTODIR;  
$SELFAUTOPARENT;
```

```
$SELFAUTOLOC/share/texmf-local/web2c;  
$SELFAUTODIR/share/texmf-local/web2c;  
$SELFAUTOPARENT/share/texmf-local/web2c;  
  
$SELFAUTOLOC/texmf-local/web2c;  
$SELFAUTODIR/texmf-local/web2c;  
$SELFAUTOPARENT/texmf-local/web2c;  
  
$SELFAUTOLOC/share/texmf/web2c;  
$SELFAUTODIR/share/texmf/web2c;  
$SELFAUTOPARENT/share/texmf/web2c;  
  
$SELFAUTOLOC/texmf/web2c;  
$SELFAUTODIR/texmf/web2c;  
$SELFAUTOPARENT/texmf/web2c;
```

找到了 texmf.cnf 一切就都顺利了。最后, 这些 TEXMF 所在的位置也可以通过环境变量设置, 而且这样设置的优先级最高。

还有, CTeX 以及它所包含 MiKTeX 不使用 web2c/kpathsea 系统, TeXLive 和 W32TeX 使用。



# 附录 C TeX 排版原理

## C.1 读取文件

TeX 的排版过程分为好几步：第一步是读取文件内容，并转换为记号列表；后面几步是对记号列表进行处理，生成出排版结果。形象地说，第一步相当于将食物嚼碎吃进胃里，后面几步相当于在胃里将嚼碎的食物消化吸收。

这里我们先来看 TeX 读取文件并进行文件内容转换的过程，即如何将文件内容分解为一串记号 (token)。例如下面的内容

```
a {brown\bf fox}
```

将被转换为如下的一串记号

$a_{11} \sqcup_{10} \{_1 b_{11} r_{11} o_{11} w_{11} n_{11} \boxed{\text{bf}} f_{11} o_{11} x_{11} \}_2$

其中，一个记号既可以是单个字符，也可以是一个控制序列 (control sequence)。

单个字符作为记号时，都带有一个类别码 (catcode)，在上面的例子中我们将类别码标在字符的右下角。左花括号 { 和右花括号 } 的类别码为 1 和 2，空格符的类别码为 10，52 个英文字母的类别码都为 11，其它可见字符的类别码是 12。

当 TeX 读到转义符 \ (类别码为 0) 时，根据该字符后面是单个非字母字符 (例如 \{) 还是多个字母字符 (例如 \bf)，它可以生成一个单字符记号，或者生成一个控制序列记号。然后，TeX 忽略后面的任意多个空格，继续处理文件。在上面的例子中，\bf 这个控制序列记号我们用方框框起来表示。而 \bf 后面的两个空格被忽略了，没有出现在记号列表中。因此这个例子输出的结果是这样的：a brown**fox**。

如果 TeX 读到一个或连续多个空行，它生成一个 \par 控制序列，表示段落的结束。如果读到一个注释符 % (类别码为 14)，它忽略本行中 % 及其后

面的所有字符，包括行尾的换行符。如果读到一个换行符（类别码为 5）或空格符，它生成一个空格符记号，并忽略紧跟其后的一个或多个空格。

## C.2 生成盒子

从文件内容读取出记号列表之后，TeX 接着开始生成盒子（box）。例如下面的文字

Yes, I'm here. Go.

将生成下面的这些盒子：

```
\hbox(6.94444+1.94444)x80.43065
.\tenrm Y
.\kern-0.83334
.\tenrm e
.\tenrm s
.\tenrm ,
.\glue 3.33333 plus 2.08331 minus 0.88889
.\tenrm I
.\tenrm '
.\tenrm m
.\glue 3.33333 plus 1.66666 minus 1.11111
.\tenrm h
.\tenrm e
.\tenrm r
.\tenrm e
.\tenrm .
.\glue 4.44444 plus 4.99997 minus 0.37036
.\tenrm G
.\tenrm o
.\tenrm .
```

要在 log 文件中记录这些信息，需要在 tex 文件中加入类似如下的命令：

```
\tracingoutput=1 \showboxbreadth=999 \showboxdepth=999
```

其中 `\showboxbreadth` 表示每个盒子里显示多少项内容，而 `\showboxdepth` 表示最多显示多少层嵌套盒子。

我们来解释这些盒子信息。首先 `\hbox` 表示这是一个水平盒子。而后边的几个数字表示该盒子的高度（height）、深度（depth）和宽度（width）分别

为 6.94444pt、1.94444pt 和 80.43065pt。高度和深度分别是基线（baseline）到盒子顶部和底部的距离，两者之和是该盒子实际占用的竖直长度。盒子信息中下面的每一行都以 `.` 开头，表示这些是包含在该盒子内部的小盒子。

每个以 `\tenrm` 开始的行表示一个由字符组成的盒子，它们的尺寸由 10pt 的 `cmr` 字体确定。

以 `\kern` 开始的行表示一个字距微调（kern），这在英文排版中是个基本的要求。在这里，相邻字母 Y 和 e 的距离稍微减少会更加美观。

以 `\glue` 开始的行表示一个粘连（glue），它描述一个弹性空隙，由 TeX 从空格符记号生成。粘连后面跟着的信息

```
3.33333 plus 2.08331 minus 0.88889
```

表示一个弹性尺寸，这三个数字分别表示自然空白，可伸长量（stretch）和可收缩量（shrink），以 pt 为单位。同样一个空格，位于单词或者不同标点符号后面，所生成的粘连尺寸各不相同，这是为了更好的排版效果。

文章排版，从本质上说，是在限定大小的各种盒子里，以尽量美观地方式放置多个子盒子。由于尺寸的限制和对齐的要求，粘连这种弹性间距就很有用处。下面的例子是盒子宽度分别为 80pt、85pt 和 90pt 时的排版结果：

Yes, I'm here. Go.

Yes, I'm here. Go.

Yes, I'm here. Go.

假设大盒子的宽度为  $w$ ，里面各小盒子，微调以及粘连的宽度之和（称为自然宽度）为  $x$ 。当  $w > x$  时，TeX 依照各个粘连的可伸长量按比例增加它们的宽度。反过来，当  $w < x$  时，TeX 依照各个粘连的可收缩量按比例减少它们的宽度；但是此时规定，各个粘连所减少的宽度不能超过它的可收缩量。粘连的实际伸长量与可伸长量的比例，或者实际收缩量与可伸缩量的比例，称为粘连调整比例，记为  $r$ 。因此，伸长时有  $0 < r < \infty$ ，而收缩时有  $0 < r \leq 1$ 。

## C.3 分段为行

经过上一步的处理，TeX 从一个段落的内容生成了一系列水平盒子，以及一些表示空白的微调和粘连。如果这些盒子的总宽度超过行宽，就需要将一个段落分成若干行。

TeX 首先考虑在粘连处断行。假设我们在某些粘连处将段落分为若干行。由于各行之间一般需要两边对齐，如上所述，我们可以适当伸缩每行内部各

个粘连，从而可以得到每行的粘连调整比例  $r$ 。TeX 中用 `badness` 值来描述各行排版的松紧程度。`badness` 值定义为

$$b = \min(100r^3, 10000)$$

另外，

$$d = (l + b)^2$$

称为该行的缺陷值 (demerit)，其中  $l$  是 `\linepenalty` 的值，默认为 10。段落总缺陷值一般等于各行的缺陷值之和。但是，如果有相邻两行松紧反差太大，总缺陷值还得再加上 `\adjdemerits` 值 (默认为 10000)。TeX 将尝试寻找使得段落总缺陷值最小的断行方式。

实际上，我们可能需要禁止或强制在某位置断行。TeX 用 `penalty` 值来刻画在某些位置断行的不适合度。例如，我们可以一些位置加入 `\penalty 100` 或 `\penalty -50` 等命令。TeX 将禁止在任何 `penalty` 大于等于 10000 的位置禁止断行，而强制在任何 `penalty` 小于等于 -10000 的位置断行<sup>1</sup>。此时，如果我们在 `penalty` 值为  $p$  的位置断行，那么该行的缺陷值的定义需要修改为<sup>2</sup>

$$d = \begin{cases} (l + b)^2 + p^2, & \text{如果 } 0 \leq p < 10000; \\ (l + b)^2 - p^2, & \text{如果 } -10000 < p < 0; \\ (l + b)^2, & \text{如果 } p \leq -10000. \end{cases}$$

这个公式需要稍作解释。当  $p \geq 10000$  时，TeX 禁止在此处断行，因此我们没必要定义缺陷值。而当  $p \leq -10000$ ，TeX 强制在此处断行，因此定义为  $(l + b)^2$  或  $(l + b)^2 - 10000^2$  对段落总缺陷的寻找毫无差别。还需注意的是，设置 `\penalty 0` 并非多余，因为它会使得 TeX 将这个位置作为断行的候选点<sup>3</sup>。

按照前面的方法，TeX 找到了一系列断点。如果此时得到的各行的 `badness` 值没超出 `\pretolerance` 值 (默认为 100)，即我们能忍受这样的排版，断行就宣告顺利完成了。否则，TeX 需要进行第二次尝试。此时，TeX 将把英文的连字算法也考虑进去，试图让各行的 `badness` 值不超出 `\tolerance` 值 (默认

<sup>1</sup>在 Plain TeX 中，宏 `\nobreak` 定义为 `\penalty 10000`，`\break` 为 `\penalty -10000`，而 `\allowbreak` 为 `\penalty 0`。

<sup>2</sup>在 TeX 中，`badness` 和 `demerit` 的定义公式看来有些随意，但在实践中却能得到良好的效果。

<sup>3</sup>在 XeTeX 中，默认无法在汉字之后断行；但利用 `\XeTeXlinebreaklocale "zh"` 命令，我们在汉字后自动添加了 `\penalty 0`，从而得到正确的断行。

为 200)。如果还是失败了，就让水平盒子溢出，并且产生 “`overfull \hbox in paragraph`” 的警告。

通过设置 `\tracingparagraphs=1`，我们可以在日志文件中记录 TeX 对段落断行时的详细信息。

## C.4 组行为页

现在我们已经将段落分为一行行，接下来的任务是将各行组成为一个个页面。因为页面的竖直长度是固定的，而每行是一个盒子，行距是一个粘连，这个任务就和断行有些类似了。

我们先看看竖直空隙的大小。首先，每个段落上面留出的空隙（即段距）由 `\parskip` 给出，默认为 `0pt plus 1pt`。其次，每行上面留出的空隙（即行间距）一般由行距 `\baselineskip` 确定，即让它的基线与上一行的基线距离为 `\baselineskip`，然后计算出行间距。但是如果这个行间距小于 `\lineskiplimit`，就改为使用 `\lineskip` 的值（这种情形在行中有较大的公式时会出现）。在 Plain TeX 中，默认的字大小为 10pt，而 `\baselineskip`、`\lineskip` 和 `\lineskiplimit` 的默认值分别为 12pt、0pt 和 1pt。这就保证各行之间在多数情形都有 2pt 的行间距，而在特殊情形也有 1pt 的行间距。我们也可以使用 `\nointerlineskip` 阻止生成某个行间距粘连，或者用 `\offinterlineskip` 阻止生成全部行间距粘连。

假定我们将若干行分到一个页面，和断行时类似，此时也可以计算出这样分页的垂直 badness 值，它的取值范围为  $0 \leq b \leq 10000$ 。不同的是，如果盒子溢出了， $b = \infty$ 。

同样，在竖直模式我们也用 `penalty` 来调整在某些位置分页的不合适程度。而且 TeX 也在 `penalty` 大于等于 10000 的位置禁止分页，在 `penalty` 小于等于 -10000 的位置强制分页。实际上，TeX 中设置了这些 `penalty` 值：

`\interlinepenalty` 在段落内部两行之间分页的 `penalty`，默认为 0；

`\clubpenalty` 在段落首行之后分页的 `penalty`，默认为 150；

`\widowpenalty` 在段落末行之前分页的 `penalty`，默认为 150；

`\displaywidowpenalty` 在行间公式末行之前分页的 `penalty`，默认为 50；

`\brokenpenalty` 在连字符行之后分页的 `penalty`，默认为 100。

利用 badness 值和 penalty 值, TeX 将计算出在某处分页的代价 (cost) 为

$$c = \begin{cases} p, & \text{如果 } b < \infty \text{ 且 } p \leq -10000; \\ b + p, & \text{如果 } b < 10000 \text{ 且 } -10000 < p < 10000; \\ 100000, & \text{如果 } b = 10000 \text{ 且 } -10000 < p < 10000; \\ \infty, & \text{如果 } b = \infty \text{ 且 } p < 10000. \end{cases}$$

万事俱备, 现在 TeX 开始分页的尝试了。和断行不同的一点是, 分页时 TeX 只考虑当前页面, 而不考虑后面怎么分页。具体来说, TeX 每次将一行加到当前页, 并计算相应的分页代价  $c$ 。这样一直下去, 直到算出的代价为  $\infty$ 。这时候, 之前各次尝试过程中, 代价最小的分页方式就是 TeX 最终的分页方式。如果有多个相等的最小值, 则选取最后一个。

如果还有页面中还有浮动对象, 分页算法还需要继续改进。这里不再详细介绍。

我们可以使用 `\tracingpages=1`, 告诉 TeX 把页面成本的计算输出到日志文件中。

# 附录 D TeX 相关软件

## D.1 纯文本编辑器

### D.1.1 TeXMaker 编辑器

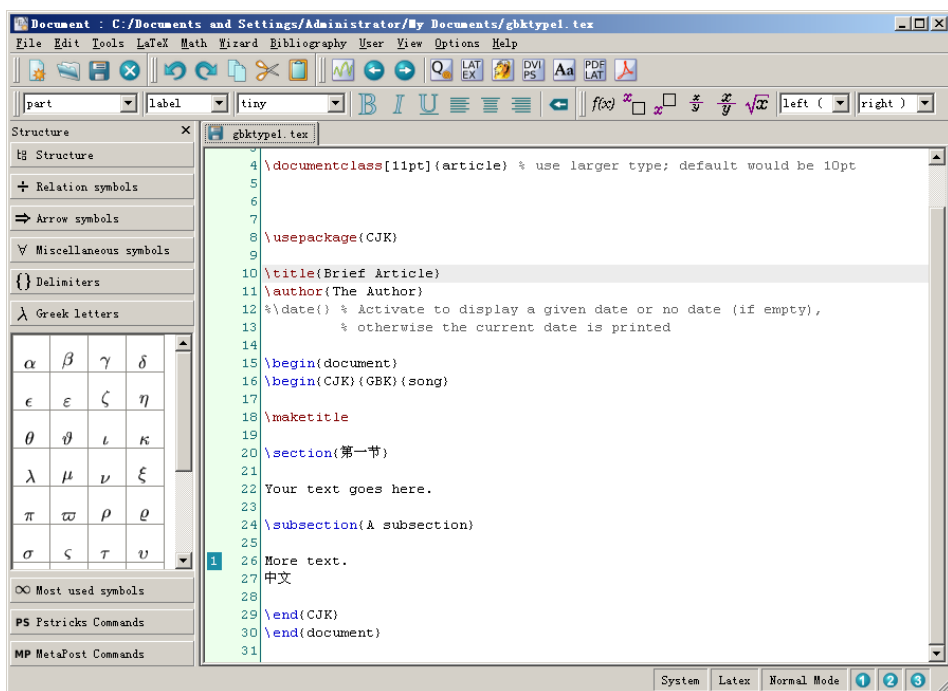


图 D.1: TeXmaker 编辑器

TeXMaker (<http://www.xmlmath.net/texmaker/>) 是法国一位中学数学

老师写的 TeX/LaTeX 编辑器，可以用它来代替 WinEdt。只要安装了 CTeX 中文套装就可以使用 TexMaker，其优点包括能够在侧边栏里包含了数十个常用数学符号，可以查看文档章节结构，以及转换 LaTeX 文件为 html 网页（图片格式）。另外，TeXMaker 在 Windows 和 Linux 下都能使用。

## D.1.2 TeXworks 编辑器

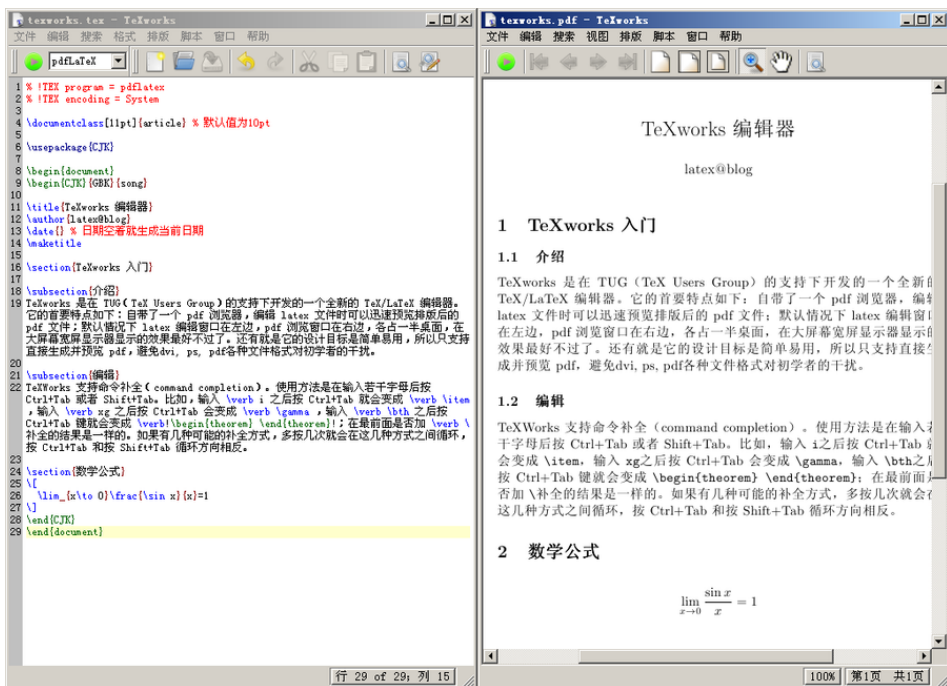


图 D.2: TeXworks 编辑器

TeXworks 是在 TUG (TeX Users Group) 的支持下开发的一个全新的 TeX/LaTeX 编辑器。它的首要特点如下：自带了一个 pdf 浏览器，编辑 latex 文件时可以迅速预览排版后的 pdf 文件；默认情况下 latex 编辑窗口在左边，pdf 浏览窗口在右边，各占一半桌面，在大屏幕宽屏显示器显示的效果最好不过了。还有就是它的设计目标是简单易用，所以只支持直接生成并预览 pdf，避免 dvi, ps, pdf 各种文件格式对初学者的干扰。

最新版本的 MiKTeX (CTeX) 已经包含了 Texworks 编辑器，可以直接



使用。TeXworks 支持通过 `synctex` 辅助文件来实现正向和反向搜索，最新的 `pdftex` 和 `xetex` 都支持在编译时生成 `synctex` 文件。在编译时设置 `synctex=0` 表示不使用，`synctex=-1` 表示使用不压缩的 `synctex` 文件，`synctex=1` 表示使用压缩的 `synctex` 文件。启用了 `synctex` 后，在 TeXWorks 的编辑窗口和浏览窗口按住 `Ctrl` 键用鼠标单击，即可在 `tex` 源码和 `pdf` 内容之间跳转（即正向和反向搜索）。

TeXWorks 支持命令补全（command completion）。使用方法是在输入若干字母后按 `Ctrl+Tab` 或者 `Shift+Tab`。比如，输入 `i` 之后按 `Ctrl+Tab` 就会变成 `\item`，输入 `xg` 之后按 `Ctrl+Tab` 会变成 `\gamma`，输入 `\bth` 之后按 `Ctrl+Tab` 键就会变成 `\begin{theorem} \end{theorem}`；在最前面是否加 `\` 补全的结果是一样的。如果有几种可能的补全方式，多按几次就会在这几种方式之间循环，按 `Ctrl+Tab` 和按 `Shift+Tab` 循环方向相反。

TeXWorks 默认使用 UTF-8 编码，如果你需要编辑用 GBK 编码的 `tex` 文件（比如 CCT 或者 CJK 文件），可以在文件的最前面加上如下一行以免乱码或者编译出错：

```
% !TEX encoding = System
```

另外，你也可以通过在文件最前面再加如下一行指定默认的编译命令：

```
% !TEX program = pdflatex
```

TeXworks 使用的是 Qt 图形界面，所以有 Windows，Linux 和 Mac OS 各种平台的版本。

### D.1.3 正向搜索与反向搜索

要想舒服地编辑 LaTeX 文件，LaTeX 编辑器和 DVI 查看程序对正向搜索和反向搜索的支持是必不可少的。

以 CTeX 套装为例，假如我们正在编辑 `docomo.tex` 文件的第 111 行，当我们点击 WinEdt 工具栏的“DVI Search”图标（或者按快捷键 `Shift+Ctrl+S`）时，DVI 查看程序 Yap 会自动运行，打开 `docomo.dvi` 文件并显示第 111 行所在的页面，这就是正向搜索；而假如我们正在用 Yap 查看 `docomo.dvi` 文件的第 9 页，当我们双击页面时，WinEdt 编辑器会自动运行，打开 `docomo.tex` 文件并跳到第 9 页所对应的 LaTeX 代码，这就是反向搜索。

在 Windows 下比较出名的 DVI 查看程序基本就只有 Yap 了，而如上所见 Yap 确实是支持正向和反向搜索的。而 LaTeX 编辑器除了 WinEdt 这个

商业软件之外，还有 TeXMaker、WinShell、TeXnicCenter、Scite、TeXworks 等免费编辑器，这些编辑器对正向和反向搜索的支持程度不一。

## D.2 可视化编辑器

### D.2.1 LyX 文档处理软件

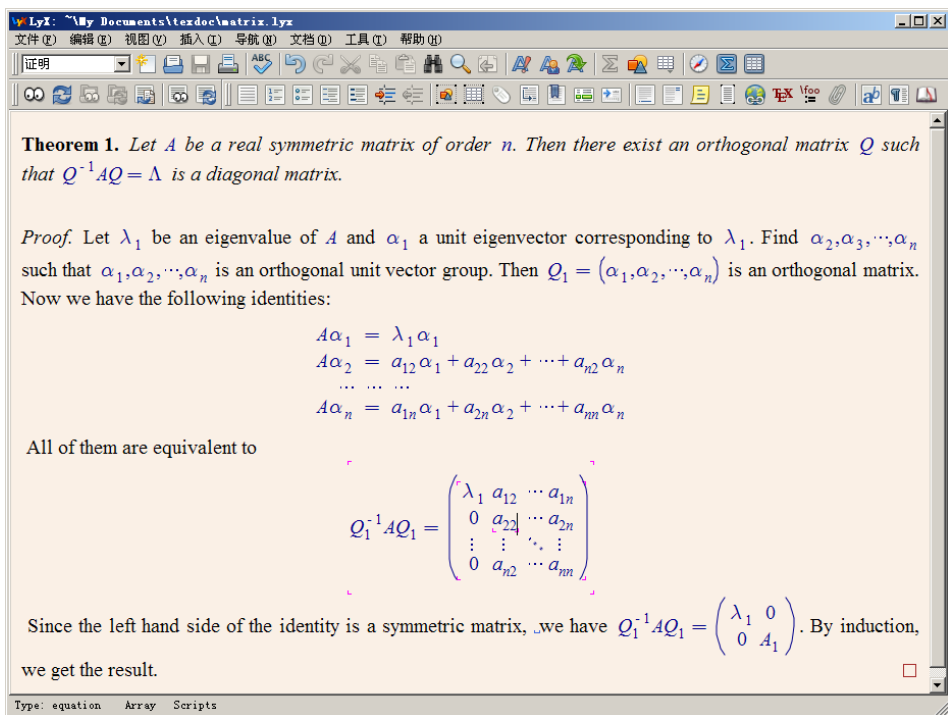


图 D.3: LyX 文档处理软件

LyX 是一个近乎所见即所得的文档处理软件，它既有着 Office 软件直观易用的优点，又能利用 TeX 后端生成高质量的科学文档。因此，它适合对 TeX/LaTeX 有一定了解的用户使用。最近发布的 LyX 1.6 版本对中文的支持程度已经足够好了，这里介绍下这个软件。

在安装 LyX 之前，确保你已经安装了 CTeX 等 TeX 套装，并将 tex 程序所在的目录加入了环境变量。Windows 版本的 LyX 安装时会自动寻找并设

置 tex 环境。

安装之后打开 LyX，新建一个空白文档就可以撰写文档了。默认的文档类别是 latex 的 article 类。这时候在程序左上角的下拉框中可以选择输入文档头（标题、作者、日期、摘要），章节标题（section、subsection），各种列表（item、enumerate）等等；在工具栏里还可以插入各种其它的内容（图像、表格、标签、引用、脚注），自然还有许许多多的数学符号。撰写过程中随时点击工具栏中的 View 图标就可以生成并打开相应的 pdf 文件。撰写完毕在“文件-> 导出”菜单中就可以将 lyx 文档导出为 tex 或者 pdf 文档。

LyX 文档的编辑过程可以完全使用键盘快捷键（而且各种快捷键都可以自己定制）。比如按 Ctrl+M 可以插入行内公式，按 Ctrl+Shift+M 可以插入单行显示公式；最优秀之处在于，在编辑公式过程中你输入 `\alpha`，`\int`，`\infty` 等数学命令加空格之后会自动转换为相应的数学符号，输入 `^` 和 `_` 之后会自动切换到上下标位置，输入 `\frac` 加空格之后会自动转换为分式的模式，等等，非常直观优雅并且方便快捷。LyX 继承了 TeX 的传统，不能直接输入多于一个的空格，要输入更多的空格或者各种不同宽度的空格，可以点击工具栏的空格图标。当然也可以使用 Ctrl+Alt+Space 插入普通宽度的空格，用 Ctrl+Shift+Space 插入小宽度的空格。在 LyX 里面按 Alt+P 然后再按如下各键可以切换到各种章节标题或正文模式中：按 0, 1 一直到 6 分别为 part, chapter, section, subsection, subsubsection, paragraph 和 subparagraph；按 T 为输入文档标题，按 Shift+A 输入文档作者；按 S 可以回到正文模式中。

LyX 支持用 CJK 排版中文文档，这需要在“文档-> 首选项”菜单的“语言”栏中设置“编码”为 Chinese (GBK)，否则生成 pdf 文档时会出错。LyX 同样支持 book, report 等文档类和 powerdot, beamer 等幻灯片类，这个需要在上述菜单的“文档 Class”栏中选择；如果你使用 beamer 排版中文文档，还需要在“Class options”处填上 cjk 选项。注意这样虽然能正常编译，但是生成的 pdf 文件的中文书签会出现乱码，据说在文档编码设置中改用 utf8 编码可以避免此问题，目前未作测试。

LyX 从 2.0 版本开始也直接支持用 XeTeX 来编译中文文档，可以按照如下步骤设置：

1. 在“文档-> 首选项”菜单里，将“语言”栏的“编码”改为“Unicode (XeTeX) (utf8)”。
2. 在“文档-> 首选项”菜单里，在“LaTeX 序”栏中加入 XeTeX 相关的中文设置，比如

```
\usepackage[cm-default]{fontspec}
```

```
\usepackage{xunicode}
\usepackage{xltextra}
\setmainfont[BoldFont=SimHei,ItalicFont=KaiTi_GB2312]{SimSun}
\setsansfont[BoldFont=SimHei]{KaiTi_GB2312}
\setmonofont{NSimSun}
\XeTeXlinebreaklocale "zh"
\XeTeXlinebreakskip = 0pt plus 1pt
```

3. 在“文档-> 首选项”菜单里，将“Output”栏的“Default output format”改为“PDF (XeTeX)”。

4. 现在就可以点击工具栏的“View”图标以预览 pdf 文档，或者点击“文件-> 导出->PDF (XeTeX)”以生成 pdf 文档。

D.2.2 TeXmacs 文档排版软件

TeXmacs (<http://www.texmacs.org>) 是比 LyX 还优秀的科学文档排版软件，因为 LyX 只是半所见即所得的，而 TeXmacs 是完全所见即所得的。在 TeXmacs 中撰写好之后可以导出为 tex 文档，也可以直接导出为 pdf 文档，不需要用 latex 程序编译。

启动 TeXmacs 后我们可以在菜单中对文档做如下这些设置：

菜单项	功能
文档-> 查看-> 非正式标记-> 在纸上显示	显示各种标志
文档-> 查看-> 页面布局-> 显示页眉页脚	显示页眉和页脚
文档-> 查看-> 页面布局-> 在纸上显示边白	显示边距
文档-> 查看-> 样式->Latex	显示 LaTeX 风格命令
文档-> 页面-> 类型->Paper	显示页码

最后我们需要选择文档的类型（如同 latex 中的 `\documentclass`），里面有 article, book, seminar 等等，一般我们用 article。现在选择“文件-> 保存”菜单项将文档保存为 TeXmacs 默认的.tm 格式，比如 xxx.tm 文件，基本的设置就完成了。

现在开始撰写文档，在菜单中可以选择插入标题、章节、环境、表格、图片、公式等等。我们只讲如何用键盘方式方便的输入内容：首先，输入 \$ 符号开始一个行间公式，按 **Ctrl+Tab** 在行间公式和独立公式之间切换，而输入 **Alt+Shift+&** 开始一个独立的多行公式，按 **Alt+Shift+\*** 在是否显示公式编号之间切换。其次，输入 \符号进入命令模式，这里你可以输入许多 latex 命

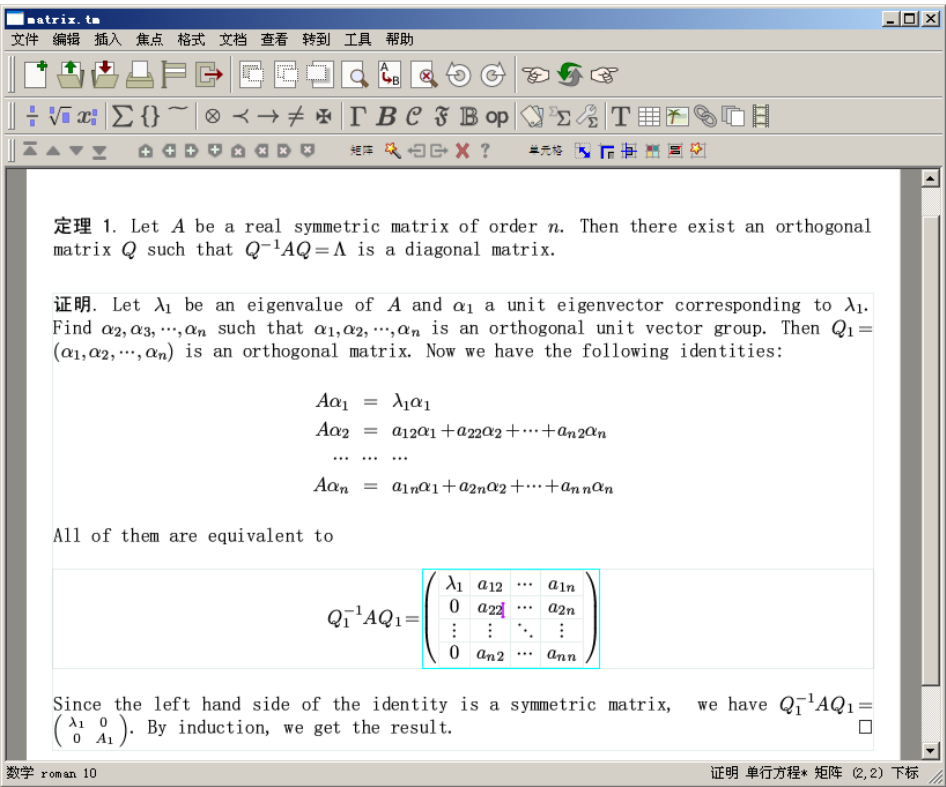


图 D.4: TeXmacs 文档排版软件

令和 texmacs 命令，然后按回车键生效；比如你在公式里面输入 `\frac` 并按回车键就生成一个分式。因此在 TeXmacs 中 `$` 和 `\` 这两个特殊符号不能直接输入，但是你可以依次输入 `\$` Enter 得到 `$` 符号，依次输入 `Shift+F5` 得到 `\` 符号。

好，现在你可以去尝试 TeXmacs 了，你会发现许多 latex 命令都是有效的，比如 `\section`，`\subsection`，`\label`，`\ref` 等等。

### D.3 新版 Office 的公式排版

微软的 Office 软件从 2007 版本开始使用新的文档格式（docx，pptx 等），从此 Word 里面对公式的支持摆脱了残废状态，使得在 Word 中不

借助 MathType 和 Aurora 等插件输入复杂数学公式成为可能。而且, 微软还特地为公式的排版定制了一个数学字体 Cambria Math, 再加上在公式的输入中可以使用许多 LaTeX 命令, 因此用微软 Word 快捷地排版美观的公式不再是天方夜谭。

这里介绍下在新版 Word 中输入公式的方法。使用键盘快捷键 “Alt+=” 或者在插入菜单中选择 “公式” 项均可插入一个新的数学公式。首先需要说明的就是, 新版 Word 中的公式和 LaTeX 中类似, 可以有行间公式和独立公式两种, 两种公式在显示上有一些区别 (比如独立公式一般自动居中)。和 LaTeX 中不同的是, 在 Word 中一个公式属于哪种类型的是由该行是否有其它内容自动决定的。也就是说, 你在某些文字后面直接按 “Alt+=” 将得到行间公式, 而新起一行后再按 “Alt+=” 则得到自动居中的独立公式。这点还是 Word 更加容易使用一点。

在 Word 公式里面大部分常用 LaTeX 命令均可使用, 比如输入 `\infty` 再按空格就得到个无穷号, 输入 `\sqrt` 再按空格就得到个根号, 输入 `\int_a^b` 再按空格就会得到个定积分。在 LaTeX 和在 Word 中数学公式的输入方式主要有如下两个区别: 一是分式的输入方式不同, 在 Word 中用更自然的方式来输入分式; 比如在 LaTeX 里面需要输入 `\frac{a+b}{c}`, 而在 Word 中只需要输入 `(a+b)/c` 再按空格就出来了。二是命令参数的括号不同, LaTeX 使用花括号 `{}` 而 Word 使用圆括号 `()`; 比如在 LaTeX 中输入 `\sum_{k=0}^{n_1}`, 而在 Word 中输入 `\sum_(k=0)^(n_1)`。其它各种命令及用法可以看微软程序员写的这个说明文档<sup>1 2</sup>。

## D.4 在网页中显示数学公式

要在网页中显示 tex 公式可以使用 jsMath 或者 ASCIIMathML。两者都是使用 JavaScript 语言来转换网页中包含的 tex 公式。jsMath 将数学公式用 tex 算法直接显示出来, 在几乎所有浏览器中都能直接显示, 但是 jsMath 包含了许多文件, 在个人网页中使用也许不太方便。而 ASCIIMathML 将数学公式转换为 MathML 公式再丢给浏览器显示, 在 IE 和 Chrome 中没法显示 (IE 安装了 mathplayer 后可以显示 MathML 公式), 但在 Firefox 中可以正常显示。估计迟早 IE 和 Chrome 会支持 MathML 公式, 因此在个人网页中用 ASCIIMathML 来显示公式比较方便。

<sup>1</sup>网页说明: <http://www.unicode.org/notes/tn28/>

<sup>2</sup>详细文档: <http://www.unicode.org/notes/tn28/UTN28-PlainTextMath-v3.pdf>

在网页中添加 ASCIIMathML 支持很容易,首先到这里下载 ASCIIMathML.js 文件并上传到网页文件夹中,然后修改网页代码,在 `<head>` 和 `</head>` 之间添加如下语句:

```
<script src="http://xxxx.js" type="text/javascript"></script>
```

其中 `http://xxxx.js` 指的是 ASCIIMathML.js 文件上传后的实际地址。然后在 Firefox 中打开网页就可以看到所有用 `$` 或者 `$$` 包含的 tex 公式已经漂亮地显示出来了。

ASCIIMathML.js 文件是可以修改以适合自己的需要的。我作了一些修改:

```
var mathcolor = ""; // 公式颜色。默认值为 "blue", 设为空表示使用文本颜色
var mathfontfamily = "Times New Roman" // 改用好看的公式字体
var translateLaTeX = false; // 网页中需要介绍 latex, 故禁用 $ 和 $$ 的识别
var translateASCIIMath = true; // 网页中使用特殊的重音符 \` 来标示公式
```

其中, ASCIIMathML 添加了使用重音符 (```) 来标示公式的方法, 重音符就是键盘上 1 左边的那个字符。这样, 以后需要直接显示 tex 代码的公式还是用 `$` 和 `$$` 来标示, 而需要处理再显示的公式就用 ``` 来标示。例如下面的公式:

```
`e^x = \sum_{n=0}^{\infty} \frac{x^n}{n!}`
```

显示了指数函数的幂级数展开式。

jsMath 现在有了继任者 MathJax。MathJax 是美国数学会等组织和公司资助的项目, 在 jsMath 的基础上开发。MathJax 同时支持 latex 和 mathml 公式的排版, 也同时支持用 html+css 和 mathml 输出排版结果。和 jsMath 一样, MathJax 的代码也是由许多部分组成的, 在个人网页上不方便直接使用, 不过它提供了 CDN 供用户使用, 只需要在网页代码中添加如下几行代码就可以了, 详情请看官方文档的介绍。

## 参考文献

- [1] Donald E. Knuth. The  $\text{\TeX}$ Book. Addison-Wesley, 1993.
- [2] 邓建松, 彭冉冉, 陈长松.  $\text{\LaTeX}$  2 $\epsilon$ 科技排版指南. 科学出版社, 2001.
- [3] 陈志杰, 赵书钦, 万福永.  $\text{\LaTeX}$  入门与提高. 高等教育出版社, 2002.
- [4] Alpha Huang.  $\text{\LaTeX}$  Notes. <http://www.dralpha.com>, 2008.
- [5] 胡伟.  $\text{\LaTeX}$  2 $\epsilon$ 完全学习手册. 清华大学出版社, 2011.



# 全书索引

- [T] \#, 8
- [T] \\$, 8
- [T] \%, 8
- [T] \&, 8
- [L] \<, 38
- [L] \), 38
- [L] \l, 38
- [L] \l, 38
- [T] \^, 8
- [T] \\_, 8
- [L] \\_, 9
- [T] \~{ }, 8
- [T] \{, 8
- [T] \}, 8
- [L] \addcontentsline, 34
- [L] \addtocounter, 58
- [L] \Alph, 57
- [L] \alph, 57
- [T] \alpha, 39
- [P] \ar, 44
- [L] \arabic, 57
- [L] \arraystretch, 31, 45
- [L] \author, 14
- [T] \baselineskip, 29, 85
- [L] \baselinestretch, 30
- [L] \begin, 8
- [T] \beta, 39
- [L] \bfseries, 32
- [L] \bibitem, 15
- [L] \bibname, 34
- [T] \Big, 41
- [T] \big, 41
- [T] \Bigg, 41
- [T] \bigg, 41
- [L] \caption, 25, 27
- [T] \cdot, 37, 40
- [L] \centering, 11
- [L] \chapter, 14
- [L] \chapter\*, 15
- [L] \cite, 15
- [P] \CJKfamily, 64
- [P] \contentslabel, 37
- [P] \contentspage, 37
- [T] \cos, 40
- [L] \date, 14
- [L] \DeclareFontEncoding, 62
- [L] \DeclareFontFamily, 62
- [L] \DeclareFontShape, 62
- [T] \Delta, 39
- [T] \delta, 39
- [L] \depth, 17
- [L] \documentclass, 8
- [P] \draw, 47
- [L] \end, 8

- [T] \eqno, 39
- [T] \eta, 39
- [P] \fangsong, 63
- [L] \fbox, 16
- [L] \fboxrule, 17
- [L] \fboxsep, 17
- [P] \filcenter, 37
- [P] \filldraw, 48
- [T] \font, 62
- [L] \footnotesize, 33
- [L] \frac, 40
- [L] \framebox, 17
- [P] \frametitle, 51
- [T] \Gamma, 39
- [T] \gamma, 39
- [P] \geometry, 34
- [L] \height, 17
- [P] \heiti, 63
- [L] \hline, 23
- [T] \hphantom, 21
- [L] \Huge, 33
- [L] \huge, 33
- [P] \hypersetup, 55
- [P] \includegraphics, 25
- [P] \includepdfmerge, 73
- [L] \index, 16
- [L] \indexname, 34
- [T] \infty, 40
- [T] \int, 40
- [P] \invisible, 53
- [T] \iota, 39
- [L] \item, 10
- [L] \itshape, 32
- [T] \jot, 44
- [P] \kaishu, 63
- [T] \kappa, 39
- [T] \Lambda, 39
- [T] \lambda, 39
- [L] \LARGE, 33
- [L] \Large, 33
- [L] \large, 33
- [T] \left, 40
- [T] \lim, 40
- [T] \lineskip, 85
- [T] \lineskiplimit, 85
- [L] \linespread, 30
- [L] \listoffigures, 15
- [L] \listoftables, 15
- [T] \ln, 40
- [L] \makebox, 17
- [L] \makeindex, 16
- [L] \maketitle, 14
- [P] \mathbb, 39
- [P] \mathcal, 39
- [L] \mathrm, 40
- [L] \mbox, 16
- [L] \mdseries, 32
- [L] \multicolumn, 24
- [L] \newcommand, 56
- [L] \newcommand\*, 56
- [L] \newcounter, 58
- [L] \newenvironment, 57
- [L] \newline, 9
- [L] \newtheorem, 12
- [P] \node, 47
- [L] \normalfont, 32
- [L] \normalsize, 33
- [T] \Omega, 39

- [T] `\omega`, 39
- [P] `\onslide`, 53
- [P] `\onslide*`, 53
- [P] `\onslide+`, 53
- [L] `\paragraph`, 14
- [L] `\paragraph*`, 15
- [L] `\parbox`, 18
- [L] `\part`, 14
- [L] `\part*`, 15
- [P] `\pause`, 50, 52
- [P] `\pdfbookmark`, 35
- [T] `\phantom`, 21
- [P] `\phantomsection`, 35
- [T] `\Phi`, 39
- [T] `\Pi`, 39
- [P] `\printindex`, 16
- [T] `\Psi`, 39
- [L] `\raggedleft`, 12
- [T] `\raggedright`, 12
- [L] `\raisebox`, 21, 26
- [P] `\reflectbox`, 22
- [L] `\refstepcounter`, 58
- [P] `\resizebox`, 22
- [T] `\right`, 40
- [T] `\Rightarrow`, 42
- [L] `\rmfamily`, 31
- [L] `\Roman`, 57
- [L] `\roman`, 57
- [P] `\rotalebox`, 23
- [L] `\rule`, 20
- [P] `\scalebox`, 22
- [L] `\scriptsize`, 33
- [L] `\scshape`, 32
- [L] `\section`, 14
- [L] `\section*`, 15
- [L] `\setcounter`, 58
- [L] `\setlength`, 29
- [P] `\setmainfont`, 66
- [P] `\setmonofont`, 66
- [P] `\setsansfont`, 66
- [L] `\sffamily`, 31
- [T] `\Sigma`, 39
- [T] `\sigma`, 39
- [T] `\sin`, 40
- [L] `\slshape`, 32
- [L] `\small`, 33
- [T] `\smash`, 22
- [P] `\songti`, 63
- [T] `\sqrt`, 40
- [L] `\stepcounter`, 58
- [P] `\stepwise`, 50
- [L] `\subparagraph`, 14
- [L] `\subparagraph*`, 15
- [L] `\subsection`, 14
- [L] `\subsection*`, 15
- [L] `\subsubsection`, 14
- [L] `\subsubsection*`, 15
- [L] `\tableofcontents`, 15
- [P] `\texorpdfstring`, 36
- [T] `\textbackslash`, 8
- [T] `\textbar`, 8
- [L] `\textbf`, 32
- [T] `\textgreater`, 8
- [L] `\textit`, 32
- [T] `\textless`, 8
- [L] `\textmd`, 32
- [L] `\textrm`, 31
- [L] `\textsc`, 32

- [L] `\textsf`, 31
- [L] `\textsl`, 32
- [L] `\texttt`, 31
- [P] `\texttwosuperior`, 36
- [L] `\textup`, 32
- [L] `\thesection`, 57
- [T] `\Theta`, 39
- [P] `\tikzset`, 47
- [L] `\tiny`, 33
- [L] `\title`, 14
- [P] `\titlecontents`, 37
- [P] `\titlerule*`, 37
- [T] `\to`, 40
- [L] `\totalheight`, 17
- [L] `\ttfamily`, 31
- [P] `\uncover`, 53
- [L] `\upshape`, 32
- [T] `\Upsilon`, 39
- [P] `\usecolortheme`, 54
- [P] `\usefonttheme`, 54
- [P] `\useinnertheme`, 54
- [P] `\useoutertheme`, 54
- [L] `\usepackage`, 8
- [P] `\usetheme`, 53
- [P] `\usetikzlibrary`, 48
- [L] `\value`, 57
- [P] `\visible`, 53
- [T] `\vphantom`, 21
- [L] `\width`, 17
- [P] `\XeTeX`, 66
- [T] `\XeTeXlinebreaklocale`, 66
- [T] `\XeTeXlinebreakskip`, 66
- [T] `\Xi`, 39
- [P] `\xymatrix`, 43
- [T] `\zeta`, 39
- [L] `abstract`, 14
- [P] `align`, 41
- [P] `align*`, 41
- [P] `aligned`, 42
- [P] `CD`, 43
- [L] `center`, 11
- [P] `CJK`, 64
- [P] `CJK*`, 55
- [L] `corollary`, 12
- [L] `description`, 10
- [L] `displaymath`, 38
- [L] `document`, 8
- [L] `enumerate`, 10
- [L] `eqnarray`, 42
- [L] `eqnarray*`, 42
- [L] `equation`, 39
- [L] `figure`, 27
- [L] `flushleft`, 11
- [L] `flushright`, 11
- [P] `frame`, 51
- [P] `gather`, 41
- [P] `gather*`, 41
- [P] `gathered`, 42
- [L] `itemize`, 10
- [L] `math`, 38
- [L] `minipage`, 18
- [P] `split`, 42
- [L] `table`, 24
- [L] `tabular`, 23
- [L] `thebibliography`, 15
- [L] `theorem`, 12
- [P] `tikzpicture`, 46