第一章　简介

§1.1　文本格式

在当今时代，计算机的一个最通用的功能就是对文本的电子化处理，它主要由如下四步组成：

1. 文本输入到计算机里，存贮起来供以后修改、扩充和删减；
2. 格式化输入文本，使其以相同长度的行和特定尺寸的页显示出来；
3. 在计算机的监视器上显示格式化后的结果；
4. 把最终的输出送到打印机上打印出来。

有很多字处理系统可以在一个软件包中同时实现这四方面的功能，因此用户也就意识不到上面这几种划分。而且，第 3 ， 4 步实际上是一样的：都是把格式化后的结果送到一个输出设备上，只不过一种是监视器，另一种是打印机罢了。

而类似于 TEX这样的文本格式化程序，就只进行第 2 步的处理。任何一种文本编辑器都可以用来输入和修改源文本。如果你已经熟悉而且习惯于使用某一编辑器，那就不妨继续用它。而字处理程序就不一定满足这里的要求了，因为这种程序通常要加入很多不可见的控制字符。对字处理程序而言，所见即所得是一个非常好的功能。但是另一方面，你所得到的，也不必就是你所见到的。

用来作为格式化程序输入的在编辑器生成的文本中，应该包含一些特殊命令或指令，但这些指令是用可以看到的普通文本表示的。从某种意义上来，这种供格式化用的指令集很像一种装饰语言，它只是用来表示段落、章节等等从哪儿开始，而不是直接对文本进行格式化处理。而在格式化的过程中，这些指令是如何被解释的，则要看所选用的版面设计格式了。同样的文本，在不同的版面格式下可以形成完全不同的样式。

而格式化程序的功能远不至这些。实际上 TEX也是一种有丰富功能的编程语言，知识丰富的用户可以用它编写代码，来增加某一功能。 LATEX自身也就是一组复杂的宏的集合。而且任何用户都可以通过编程对 LATEX进行扩展，或者直接利用其它程序员已设计好的宏。 TEX和 LATEX的功能并不只是那些包含在基本软件包中的内容。

对于格式化软件而言，文本处理的最后一步是把结果送到输出设备上，即打印机或者计算机监视器上，甚至可以是一个文件。实现这种功能的程序称为驱动程序；它把格式化程序已编码好的输出翻译成用户可以使用的某一设备上的特定指令。这也就是说，对每种类型的打印机，也就必须有相应的驱动程序。

§1.2 TEX与其演化史

对于科技著作而言，在能排版出如同书籍一样漂亮的格式化程序中，功能最强的就是 Donald E. Knuth 所设计的 TEX程序了，其名字是由希腊字母 $\tau\epsilon\chi$ 的大写形式组成的。正是由于这个原因，其最后一个字母的发音并不是 x，而类似于苏格兰语单词 loch 或者德语单词 ach 中的 ch，也类似于西班牙语中的 j 或俄语中的 kh。这个名字强调指出了数学公式的印刷是该程序的不可分割的一部分，而不是额外附加上去的。除了 TEX外，Knuth 还设计了另一个软件 METAFONT，用来生成各种字符字体。在标准的 TEX软件包中有 75 种不同设计尺寸的字体，而且每种字体有八种不同的放缩比例。所有这些字体都是用 METAFONT程序生成的。为了满足其它应用的需要，还设计了其它字符字体，如古斯拉夫语或日语字母的字体，有这些字母的文本也可以用书籍质量排版出来。

§1.2.1 TEX程序

最基本的 TEX程序是由一些很基本的命令组成，它们可以完成简单的排版操作和程序设计功能。然而，TEX也允许用这些基本命令定义一些更复杂的高级命令。这样就可以利用低级的结构块，形成一个用户界面相当友好的环境。

当处理器运行 TEX时，该程序首先读取所谓的格式文件，格式文件中包含各种以基本语言写成的高级命令，也包含分割单词的连字号安排式样。接着处理程序就处理源文件，源文件由要处理的真正文本，以及在格式文件中已定义了的各种命令组成。

创建新格式也是一件需要由知识丰富的程序员来做的事情。把定义写到一个源文件中，这个文件接着被一个名叫 initex的特殊版本 TEX程序处理。它采用一种紧凑的方式存贮这些新格式，这样就可以被通常 TEX程序很快地读取。

虽然一般用户可能从来用不着编写这格的格式文件，但是提供给用户的有可能是需要用 initex来安装的格式源文件。例如，当我们要更新 LATEX 2_ε格式时，我们就要按照第 D.4 节所描述的方法进行操作。

§1.2.2 Plain TEX

Knuth 设计了一个名叫 Plain TEX的基本格式，以便与低层次的 TEX互应。这种格式是 TEX字处理的相当基本的部分，以致于我们有时候根本分不清到底哪是真正的 TEX处理程序，哪是这个特殊的格式。大多数声称只使用 TEX的人，实际上指的是只用 Plain TEX。

Plain TEX也是其它高极格式的基础，这些格式进一步加强了把 TEX和 Plain TEX认为是同一事物的印象。

§1.2.3 LATEX

Plain TEX的重点还只是停留在如何排版的层次上，而不是从一位作者的观点来看问题。当然对 TEX深层功能的进一步发掘，需要相当高超的编程技巧。因此它的应用就需要高级排版和程序设计人员。

正是由于这种原因，美国计算机学家 Leslie Lamport 开发了 LATEX格式，这种格式提供了一组生成复杂文档所需要的更高级命令。利用这种格式，即使使用者没有排版和程序设计的知识也可以充分发挥由 TEX所提供的强大功能，能在几天，甚至几小时内生成大量具有书籍印刷质量的结果。在生成复杂表格和数学公式方面，这一点表现得尤为突出。

LATEX相对于其基础 Plain TEX而言，更像一个包装语言。它可以在作者根本不知道所以然的情况下，自动给出标题，章节，表格目录，交叉索引，公式编号，文献引用，浮动图表。版面布局信息包含在类文件中，这些类文件并不是位于源文件中的。这些布局可以改动，也可以直接套用。

由于 LATEX是在二十世纪八十年代出现的，同其它软件一样，它也周期性地进行更新和修订。经过了很多年，版本号固定为 2.09 ，而修订只是用是日期来区分。较近的一次大修订发生在 1991 年 12 月 1 日，其后直到 1992 年 3 月 25 日为止，还有几次小的修订。

§1.2.4 LATEX 2ε

由于 LATEX相当普及，以及它在许多原本没有想像到的领域中的扩展，再加上计算机技术的日新月异，特别是价格低廉，但功能强大的激光打印机的出现，使得相当广泛的一类格式都冠以 LATEX的标签。为了尝试建立一个真正的改进标准，在 1989 年 Leslie Lamport, Frank Mittelbach, Chris Rowley 和 Rainer Schöpf 创立了 LATEX3 项目。他们的目标是建立一个最优的，有效的命令集合，这些命令是来自于各种软件包为了实现某一目的而设计的。

正如项目名称所表明的，它的目标就是得到 LATEX的一个新版本 3 。然而，由于这是一个长期目标，朝向这个目标迈进的第一步就是在 1994 年中发行了 LATEX 2ε并出版了 Lamport 基本手册第二版，同时还有一本新书，专门描述在新系统中许多可用的扩展软件包和 LATEX程序设计。 LATEX 2ε是在本世纪末左右出现的令人著目的 LATEX3 之前的现在标准版本。

实际上，在 LATEX 2ε出现之前，其处理字体安装和选择的一些部分已经以新字体选择框架 (或 NFSS) 的形式公开了，而且被许多组织或个人集成到其软件中。这种框架有两个版本，但不幸的是它们并不兼容，两个版本分别相应于 LATEX2.09 和 LATEX 2ε 。后来以一种完全与 2.09 版本兼容的方式对 NFSS进行了重新实现。

§1.2.5 在不同类型计算机上的 TEX

TEX和 LATEX原本是设计运行在中央大型机上的，但是随着工作站，甚至更强大的个人计算机 (即 PC 机) 的发展，包括 LATEX在内的 TEX软件包已经可以运行在强大的小型机上了。在本书中有关编写 LATEX文本文件的所有内容都同样地适用于大型机和个人计算机。

这样就对使用者提出了一个问题。在中央大型机上有操作员负责管理 TEX软件及附件的安装与维护。然而，工作站或 PC 机的用户也就需要同时客串管理员的角色，即他 (她) 必须比以前的 TEX用户要多知道一些关于如何设置 TEX和 LATEX的东西。或者更经常发生的情形是，他们必须知道是谁帮他们做这些事。这个人通常也就称为小团体里的 TEX专家 (TeXGuru) 。

对于工作站和 PC 机而言，相当实用的输出方式是屏幕预览，即把输出显示在监视器上。在进行实际打印之前，尤其是输出中有复杂的表格或数学公式时，这是一种相当经济的检查页面排版如何的方法。本书中练习的结果最好就只在监视器上查看一下就可以了。

在 PC 机或工作站上也有 Shell，它使用户可以只需按一下按钮或点击一下鼠标，就可以在编辑器和处理程序之间来回切换。对于这样的系统，TEX实际上也是所见即所得了。

§1.3 LATEX 2ε的新内容

本节内容主要针对于那些已相当熟悉 LATEX2.09 的读者。下面简要列出了 LATEX 2ε的新功能。

§1.3.1 类与宏包

在 LATEX2.09 与 LATEX 2ε之间的一个最本质的差别就是声明所有布局的第一条命令。这个差别实际上使得这两个版本可以兼容。

在 LATEX2.09 中，必须像下面这样来声明所需要的主样式，这个样式同时带有一些选项：

```
\documentstyle[ifthen,12pt,titlepage]{article}
```

这里的主样式是 article，它保存在一个叫 article.sty 的文件中，而同时用 12pt 作为基本字体大小，标题放在单独一页上。有上百个这样的附加文件可以用来做为样式选项 (也称为子样式)。

然而， LATEX 2ε对主样式的补充选项与真正的内部选项之间有一个明确的区分。现在主样式更名为类，而扩展文件称为宏包。上面那个初始化声明现在变为

```
\documentclass[12pt,titlepage]{article}
\usepackage{ifthen}
```

布局信息包含在文件 article.cls 中，它可以处理选项 12pt 和 titlepage 。而文件 ifthen.sty 还像以前那样读取；然而，这里的新功能是它也可以有自己的内部选项。不仅如此，那些列在 \documentclass 命令中的选项，由于被看作是全局选项，因此对所有宏包都有作用（见 3.1.2 节）。

初始化声明 \documentstyle 在 LaTeX 2ε中还可以使用，这时它切换到一个兼容模式，来模拟 LaTeX2.09 中的行为。

为了帮助那些坚韧不拔的 LaTeX程序员更好地进行程序设计，现在增加了许多新功能。对选项的处理做了改进，而且如上所述，也可以在宏包中加进选项。增加了一些安全机制，以保证版本号的匹配。在读其它文件时，现在有了更好的测试方法，以保证该文件不存在时，采取其它的方法。在附录 C 中对这些程序设计要素进行了描述。

§1.3.2 字体管理

在 LaTeX2.09 中， TeX的计算机现代字体 (Computer Modern Fonts) 被牢靠地固化在格式中。在人们喜欢用的字体也就那么几种的年代里，这不失为一种可行的方法。但到了今天，可用的字体数目繁多，特别是 PostScript 打印机的出现，更加要求一个有弹性的系统。新字体选择框架 (NFSS) 应运而生，并与 LaTeX 2ε完全结合为一体。要选择非计算机现代字体作为基本字体只是一些很简单的重定义（见 8.5 节）。

NFSS也改变了在文档内部引进字体的方法。 LaTeX2.09 继承了 TeX的字体命令，如 \bf（黑体）， \it（斜体）都是严格地选择一特定的字体。这些命令中只有字体大小维持不变。而在 NFSS中，字体是用某种属性来描述的，可以分别彼此独立地进行选择。因此就有可能先选择黑体，然后选斜体，从而得到黑斜体，而这在 LaTeX2.09 中是行不通的。

LaTeX 2ε鼓励使用字体选择命令，而不是用字体声明。例如，为了强调某一文字，命令 \emph{word} 就比声明 {\em word} 要好。这样的命令对初学者来说，更符合逻辑，虽然习惯于用后者的经验丰富的 LaTeX2.09 用户可能持相左的看法。

在数学模式中，文本的字体是通过用特殊的数学字母命令来选择的，而不是原来的字体声明。也就是说，在数学模式中不允许原先的 \rm, \bf 和 \cal 等声明，而代之以 \mathrm, \mathbf 和 \mathcal 这些有参数的命令。

§1.3.3 浮动对象的安排

在 LaTeX2.09 中一个令人头痛的问题就是如何安排浮动对象（图与表），使它出现在人们最希望看到的地方。而浮动对象的安排有一套相当复杂的规则，并不是所有的人都能很好掌握。 LaTeX 2ε提供了两种新的机制，以控制这一过程，其中一种是不鼓励对浮动对象的安排，另一种则鼓励这种安排。

通过使用命令 \suppressfloats 可以使当前页中没有浮动对象。作为选项，可以使用参数 t 或 b 来只禁止浮动对象不出现在当前页面的顶部或底部。

另一方面，有一个新的浮动位置指定符 !，它可以取消这个浮动对象所在页的关于可以出现的浮动对象数与文本总量的限制。这也克服了通常会出现的一种不满，而原来必须重定义 \texttraction 或其它浮动安排参数，并进行无数次的调试才能缓解。浮动位置指定符 ! 同其它参数一样使用：如，

```
\begin{figure}[!]
```

现在为了使浮动对象与文本更好的分离开，也可以在其顶部或底部画上定义好的标尺。在 6.6 节中对这些功能进行了介绍。

§1.3.4 扩充的语法

与 2.09 版本相比，在 LaTeX 2ε中对几条命令的语法进行了扩充。当然原来的语法仍然有效。

- \newcommand, \renewcommand, \newenvironment 和 \renewvironment 命令用来定义新的命令和环境，其除了几个必须的参数外，还可以包含一个可省略的参数（ 7.3 和 7.4 节）。
- 盒子命令 \makebox, \framebox 和 \savebox 在定义其实际尺寸时，可以引用其自然的尺度。也就是说，你可以用定义其宽度为自然宽度的两倍。见 4.7 节。
- 现在 \parbox 和 minipage 环境除了定义水平宽度外，还可以指定一个竖直高度。有一个新的内部安排参数来决定盒子中的文本是否需要被推向顶部，居中，或者放在底部，甚至伸展文本以填满整个盒子（ 4.7.5 节）。
- 以前命令 \settowidth 可以用来测量某些文本的宽度；现在又补充了 \settoheight 和 \settodepth（ 7.2 节）。

§1.3.5 与 LaTeX2.09 的兼容性

为了使 LaTeX 2ε与 LaTeX2.09 尽可能地保持兼容性，已做了各种各样的努力。这就是指根据原来版本标准写的文档，在新的版本下应得到相同的结果。同时也表明软件包中绝大多数的样式（或子样式）选项应具有与以前一样的功能，而不需任何修正。

这只是理论上应该如此。实际上，一定存在着不兼容之处。那么用户遇到这种情形的状况是怎样的呢？

- 如果只使用高级命令，在命令中不包含 @ 字符，那么兼容性是 100% 。
- 如果运用了内部命令，这就可能会导致问题，特别是如果其中包含盒子或者输出程序。
- 如果使用了内部字体控制命令，特别是那些来自于 lfonts.tex 文件的命令，就很有可能会出现问题。然而，应该指出是，即使这样，也比 NFSS的

第一个版本与第二个版本之间的不兼容性小。

据我们目前的经验，我们发现即使把盒子和输出程序算在内，也只有很少几个宏包不能在 LaTeX 2ε下运行。我们遇到的最糟糕情形是在 LaTeX2.09 下显式地调入一个支持文件，而这个文件在 LaTeX 2ε下却不保证存在。如果它被强行调入，会导致出现严重的错误信息。

§1.3.6 “我应该用哪一个版本呢？”

如果你不知道在你所用的系统中是否安装了 LaTeX 2ε，那么有许多方法可以检查这一点。在任一 LaTeX作业开始处，都会在监视器和抄本文件 (transcript file) 中列出格式的名字和日期，如：

若是 LaTeX2.09，则为 `LaTeX Version 2.09 <25 March 1992>`，

若是 LaTeX 2ε，则是 `\LaTeXe <1994/06/01>`

当然你的日期可以与这里的不同。

如果不喜欢这种方法，还可以尝试处理一个文件，第一行用的是

```
\documentclass.
```

如果你得到错误信息：

```
! Undefined command sequence
      \documentclass
```

那么你用的就不是 LaTeX 2ε。

§1.3.7 更新 LaTeX 2ε

按计划在每年的 6 月和 12 月份要对 LaTeX进行更新，不仅要改进内部编码，也增加一些额外的功能。这就是说随着时间的推移，仅仅只说一个文档是在 LaTeX 2ε下写的还是不够的，而是不但要指明必要的版本，还要加上发行的日期。

在本书的正文和附录 ?? 的命令汇总中，对于某一不是 1994 年 6 月 1 日所发行的正式版本中命令，都标上发行日期。

也可以在文档文件中声明一个最早的可以处理所用全部功能的版本。这可以在靠近开头的地方，加上一条鉴别命令（ C.2.1 节)。对于要用的第一个版本，应该用

```
NeedsTeXFormat{LaTeX2e}[1994/06/01]
```

这里的日期必须是数字，格式要如上面的一样，即 / 年 / 月 / 日，中间有斜线和必要的零。如果有这一命令的文件被 LaTeX更早的版本处理，就会显示一条警告信息。

§1.4 如何使用本书

本指导是教科书与参考手册的混合体。它解释了 LaTeX所有的基本组成，

特别是那些新标准 LaTeX 2ε中的部分，而且尽可能地只限于那些在标准发行中的内容。与 Lamport 的书相比，这里讲得更详细些，给出了更多的例子和习题，描述了很多来自于作者经验的 “技巧”。与 The LaTeX Companion 一书不同，我们并没有对在标准发行版本以外的可用的许多宏包进行讨论，因此对普通读者而言，这些软件包可能并不是很容易得到。唯一的，也是必需的一个例外是附录 D，那里讲述了一些额外的功能。

本书适用于那些具有很少，甚至没有计算机使用经验的用户。但是它并没有包含与计算机系统有关的操作，如怎样注册用户，启动编辑器，或者使用编辑器等等。

本书的正文有相当多的重复，特别是前半部分，因此读者只要掌握了一些表达中的主要定义，并不必要一定精确在几页后给出的完整定义。另外，读者应习惯从 2.1–2.4 节开始出现的那种描述基本组成的方式。

作者已尽力避免使用计算机行话，虽然这不一定很成功。希望类似于文件和编辑器等这样再不可能用其它方式表达的术语，能广为人知。

在描述命令语法时，对那些必须精确照原样输入的部分用打字机字体表示，而那些可以改变的部分或者文本自身，则用斜体表示。例如，生成表格的命令就用如下方式陈述：

```
\begin{tabular}{ 列格式 }  文本行  \end{tabular}
```

用打字机字体表示的部分是不能省略的，而 *列格式* 表示必须在此处加上列格式的定义。可允许取的值及其组合在命令的描述中有详细的介绍。在上面的例子中，*文本行* 代表表格中的行元素，它是文本的一部分。

§1.4.1 从 2.09 中区分开 LaTeX 2ε

由于现在的标准是 LaTeX 2ε，所以这里给出的语法和功能描述都是针对于这一版本的。然而，我们相信还会有很多用户在未来的几年内仍然使用的是 2.09 版本，这一方面可能是由于他们没有机会更新，另一方面也可能因为他们本来就不想更新。因此本书也考虑到了一点。

由于这个原因，新出现在 LaTeX 2ε中的命令，或者已存在的命令，但进行了扩充，我们有记号 $\boxed{2\varepsilon}$表示。类似地，任一只适用于 LaTeX2.09 版本的文本，则标上 $\boxed{2.09}$。

由于 LaTeX 2ε是向下兼容于 LaTeX2.09 的，因此不必标出只属于原来版本的任一命令。不但如此，有些命令只所以还保留下来（如 `\bf` 和 `\rm`），主要是为了兼容，因此在 LaTeX 2ε中并不鼓励使用这些命令，而在 LaTeX2.09 中它们则是不可缺少的。

§1.5 LaTeX文件的基础知识

§1.5.1 文本与命令

每一文本都是由字符组成，这些字符放在一起组成单词。由多个单词组成句子，句子再组成段落。而段落可以做为更大单位（如章节）的一部分。

单词由一个或多个字符组成，由空白或回车终止。TeX把空白和回车都做为单词的结束符。而单词间的空白多少则无关紧要，多个空白对单词间的间隔没有更多的作用。

段落的分隔用的是一个或多个空行。同样地，段落间隔与空行的数目无关。TeX把段落中的所有单词当做一个单词长串来处理，选择其间隔，以使得尽可能得均匀，而且每一行都要向左或右调整。行的断开是自动的，与文本的输入基本无关。

行的间隔与所选的字体尺寸有关。段落是以首行的缩进或者增大的行间隔为标志的。在必要的时候，段和行的间隔都会发生细微的变化。TeX（和LaTeX）通过调整这些间隔，使得一页的顶部和底部位于需要之处。这个位置在文档内可以改变。当断行结束后，会自动地进行分页。

在最简单的情形中，一个文本文件中只有输入文本。TeX就会用标准的宽度和高度来处理这一文本；也就是说，把要排版的文本均匀分成行，段，页。

然而，每个LaTeX文档通常都不只是包含要处理的文本，还包含定义如何处理它的命令。因此就有必要给一种方法，区分开这些命令和文本。命令由不能做为文本字符的单个字符组成，或者由单词组成，它们前面都要加上一个特殊的字符，即反斜杠 \ 。

§1.5.2 LaTeX文档的结构

在每一个LaTeX文件中，都一定有导言 (preamble) 与正文 (body) 。

导言是一组命令的集合，它指定处理后面文本的全局参数，如页面格式，文本的高度和宽度，输出页的页码、页眉与页脚的组成。即使最简单的情形，导言也必须包含命令 `\documentclass` ，以指定文档的全局处理类型。这通常也是导言中的第一条命令。

如果在导言中再没有其它命令，LaTeX就会为行宽，页边，段落间隔，页面高度与宽度和其它东西选择标准值。在原始版本中，这些设置采取的是美国标准。对于欧洲用户所用的程序，存在内建的选项，使得文本高度和宽度为 A4 纸标准。而且还存在与语言有关的软件包，已把 'Chapter' 和 'Abstract' 的标题进行了翻译。

导言是用 `\begin{document}` 来表示结束的。紧接这条命令的每一样东西都被解释为正文。它由文本中混杂其它的命令组成。与导言的内容相比，

这些命令只有局部的作用，即它们只适用于一部分文本，如缩进，公式，对字体的暂时改变，等等。正文是用 \end{document} 来结束的。这通常也是文件的结束。

一个 LaTeX文件的通常语法如下：

2ε \documentclass[选项]{ 类 }

其它全局命令和定义

\begin{document}

文本与只有局部作用的命令的混合

\end{document}

可以出现在 \documentclass 命令中的所用可能的选项和类在 3.1.1 节介绍。

对于 LaTeX2.09 ，或者与它兼容，但出现在 LaTeX 2_ε之前版本，必须用 \documentsytle 来取代初始化命令 \documentclass 。在版本 2.09 中其一般性语法应该是：

2.09 \documentstyle[选项]{ 类 }

. . . . . .

\begin{document}

. . . . . .

\end{document}

在一个计算中心，所有可以使用的 LaTeX类和软件包的信息是在一本叫“局部指南” 的书中。其中也应有如何进行程序设计的介绍，以及安装了哪些输出设备，如打印机，缩影胶片，绘图仪，等等，同时这指南应该介绍了如何使用这些设备。设备驱动程序也可以识别选项，在生成输出时进行各种不连续的放缩。

§1.5.3 LaTeX的处理模式

在处理过程中， LaTeX总是处于下面三种模式之一：

1. 段落模式，
2. 数学模式，
3. 从左到右 (LR) 模式。

段落模式也就是正常的处理模式，这时 LaTeX把输入文本做为一队要被（自动）断开成行、段落与页的单词和句子。

当遇到特定命令，表示下面的文本代表公式时， LaTeX就会切换进入数学模式，在公式中空白被忽略。 is 和 i s 这两组文本都解释成 i 与 t 的乘积 it 。当遇到相应的表示公式结束的命令时， LaTeX就会切换回段落模式。

LR 模式类似于段落模式，这时 LaTeX从左到右处理输入文本，把它们做为一组不能被断行的单词来看待。例如，当普通文本被嵌入到公式中时，或

者用命令 \mbox{短文本} 来强迫 短文本 位于一行上时，LaTeX就位于 LR 模式。

理解与识别处理模式是相当重要的，因为有些命令只能用在特定的模式中，或者在不同的模式中有不同的作用。

今后我们将把段落模式和 LR 模式合称为文本模式，以表明它们的性质一致性。但要把它们与数学模式区分开，因为通常它们的差别是很大的。

§1.5.4 生成 LaTeX文档

从输入文本到在打印机上得到输出的 LaTeX文档结果，是一个三步的过程。首先利用计算机的编辑器创建（或修改）文本文件。这个文本文件由实际的文件混杂 LaTeX命令组成。

文本文件的全名由基本名加上扩展名 .tex 组成（如 sample.tex ）。计算机操作系统通常对文件名的选取有一些附加限制，如基本名和扩展名可以拥有的最多字符数，或者哪些字母不可以使用。例如，如果基本名只限于不超过 8 个字符，那么 finances.tex 就是允许的名字，而 financial.tex 则不可以做文件名。

然后文本文件由 LaTeX处理。在 1.2.1 节中已做了介绍，这时用 LaTeX格式执行 TeX程序。在大多数安装版本中，对此都有一个特定的缩略命令，叫 latex ，这样只要在它后面接下文本文件名字就可以了，不必带后缀 .tex 。

例如，如果文本文件的名字是 sample.tex ，那么应该用下面的调用来启动 LaTeX处理程序：

```
latex sample
```

在这一处理过程中，终端监视器会显示页号以及可能会有的错误和警告信息。第 9 章对这些错误信息及其后果做了介绍。当 LaTeX结束了这一处理过程后，它会生成一个新的文件，其基本名不变，后缀为 .dvi 。在上面这个例子中，它就是 sample.dvi 。

这个 .dvi（与设备无关）文件由格式化后的文本以及所需要的字符字体有关的信息组成，但是它与要使用的打印机的特征无关。这样的与设备无关的文件也叫做元文件 (metafile) 。

最后，在 .dvi 元文件中的信息要被转化成可以在选定打印机上输出的形式，这一过程是由一个叫打印机驱动程序来完成的，它与打印机是相关的。对驱动程序的调用，与调用 LaTeX程序一样，也只需使用基本名，即不需加扩展名 .dvi 。打印机驱动程序处理完后，就又会生成另一个相同基本名的文件，而扩展名与打印机有关。

例如，如果输出要关到激光打印机上，可以如下调用驱动程序：

```
dvilj sample
```

这样就会生成一个叫 sample.lj 的文件，或者输出直接就送到了打印机上。

除了 PostScript 打印机外，也存在其它高质量的点阵打印机。然而，本书的目的不是描述 DVI 驱动程序，因为在这一方面有许多可用的程序，而且有不同的名字和特征，例如可以只打印特定的页或者反序打印。在一个计算中心中，应该已安装了一些可以使用的重要的驱动程序。关于这些调用的汇总或进一步的信息可以查看任何用户都可以使用的 TEX或 LATEX局部指南。然而关于这些信息的最好来源，我们前面已提到过，那就是问那些专家了。