用于类和包作者的 LaTeX 2ε

Copyright © 1995–2006 LaTeX 项目
版权所有

赣医一附院神经科　黄旭华　　翻译

2006 年 2 月 15 日

目 录

1 介绍

本文档介绍了如何编写 LaTeX 的类 (classes) 和包 (packages)，并特别关注了如何将现有的 LaTeX 2.09 包升级到 LaTeX 2ε 包。约翰内斯 • 布拉姆斯 (Johannes Braams) 在 TUGboat 15.3 上发表的一篇文章也提到了后一个主题。

1.1 为 LaTeX 2ε 编写类和包

LaTeX 是一个文档准备系统 (document preparation system)，它使文档编写者能够专注于他们的文本内容 (contents)，而无需过多地考虑文本的格式 (format)。例如，章 (chapters) 由 `\chapter{`⟨*title*⟩`}` 表示，而不是通过选择 18pt 粗体表示。

包含如何将逻辑结构 (logical structure)(如“`\chapter`”) 转换为格式 (如“18 磅粗体，右对齐”) 的信息的文件是一个*文档类* (*document class*)。此外，一些特性 (features)(如颜色或包含的图形) 独立于文档类 (document class)，这些特性包含在*宏包* (*packages*) 中。

LaTeX 2.09 和 LaTeX 2ε 之间最大的区别之一是用于编写包和类的命令 (commands)。在 LaTeX 2.09 中，很少支持写入 `.sty` 文件，因此编写者不得不使用低级命令 (low-level commands)。

LaTeX 2ε 提供了用于构建包的高级命令 (high-level commands)。在此之上构建类和包也就容易得多，例如根据 `article` 编写本地技术报告类 (local technical report class) `cetechr` (针对化工部门)。

1.2 概述

本文档概述了如何为 LaTeX 编写类和包。它没有介绍编写包所需的所有命令：这些命令可以在 *LaTeX: A Document Preparation System*【《LaTeX：一个文档准备系统》】或 *The LaTeX Companion*【《LaTeX 指南》】中找到。但它确实描述了用于构造类和包的新命令。

第 **2.7** 节　包含一些关于编写类和包的一般建议。它描述了类和包之间的差异、命令的命名约定 (command naming conventions)、`doc` 和 `docstrip` 的使用、TeX 的原语文件 (primitive file) 和盒子命令 (box commands) 如何与 LaTeX 交互。它还包含一些关于一般 LaTeX 样式 (style) 的提示。

第 **3** 节　描述类和包的结构 (structure)。这包括在其他类和包之上构建类和包、声明选项 (declaring options) 和声明命令 (declaring commands)。它还包含示例类 (example classes)。

第 **4** 节　列出新的类和包命令。

第 **6** 节　就如何将现有的 LaTeX 2.09 类和包升级到 LaTeX 2$_\varepsilon$ 给出了详细建议。

1.3　更多信息

对于一般性的介绍 (general introduction)，包括 LaTeX 2$_\varepsilon$ 的新特性 (new features)，您应该阅读 Leslie Lamport (莱斯利 • 兰波特) [2] 的 *LaTeX: A Document Preparation System*　【《LaTeX：一个文档准备系统》】。

Frank Mittelbach (弗兰克 • 米特巴赫) 和 Michel Goossens (米歇尔 • 古森斯) [3] 的 *The LaTeX Companion, second edition*【《LaTeX 指南》，第二版】对 LaTeX 的新特性 (new features) 进行了更详细的描述，包括 200 多个宏包和近 1000 个准备好了可以运行的示例的概述。

LaTeX 系统基于 TeX，这在 Donald E.Knuth (高德纳) [1] 的 *The TeXbook*【《特可爱原本》】中有描述。

每一份 LaTeX 副本都有许多文档文件 (documentation files)。LaTeX 每六个月发布一个版本，同时会发布 *LaTeX 更新* (*LaTeX News*)，这可以在 `ltnews*.tex` 文件中找到。作者指南 (author's guide) *LaTeX 2$_\varepsilon$ for Authors*【《LaTeX 2$_\varepsilon$ 作者指南》】为作者介绍了新的 LaTeX 文档特性，它位于 `usrguide.tex` 中。指南 *LaTeX 2$_\varepsilon$ Font Selection*【《LaTeX 2$_\varepsilon$ 的字体选择》】描述了类和包作者的 LaTeX 字体择方案 (font selection scheme)，它在 `fntguide.tex` 中。`cfgguide.tex` 中的指南 *Configuration options for LaTeX 2$_\varepsilon$*【《LaTeX 2$_\varepsilon$ 的配置选项指南》】介绍了 LaTeX 的配置。而我们修改 LaTeX 背后的理念在 `modguide.tex` 中的 *Modifying LaTeX*【《修改 LaTeX》】进行了描述。

文档化的源代码 (documented source code)(来自用于生成内核格式文件的文件，生成内核格式文件是通过 `latex.ltx`) 现在可作为 *The LaTeX 2$_\varepsilon$ Sources*【《LaTeX 2$_\varepsilon$ 源代码》】。这个非常大的文档还包括一个 LaTeX 命令的索引 (index)。它可以从 `base` 目录中的 LaTeX 文件 `source2e.tex` 进行排版；这使用文档类文件 `ltxdoc.cls`。

欲了解更多关于 TeX 和 LaTeX 的信息，请联系您当地的 TeX 用户组或国际 TeX 用户组。地址及其他详情，请浏览：

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

1.4 标准类中的策略

我们收到的许多关于标准类 (standard classes) 的问题报告 (problem reports) 都与 bug 无关，但或多或少有礼貌地暗示其中包含的设计决策 (design decisions)“不是最优的”，并要求我们对其进行修改。

我们不应该对这些文件进行这样的更改有几个原因：

- 无论被误导了多少，当前的行为 (current behaviour) 显然是设计这些类时的初衷。
- 改变“标准类 (standard classes)”的这些方面并不是一个好做法，因为许多人会依赖它们。

因此，我们决定甚至不考虑进行此类修改，也不花时间为该决定辩护。这并不意味着我们不同意在这些类的设计中存在许多缺陷，但我们有许多任务比不断解释为什么 LATEX 的标准类不能更改更重要。

当然，我们欢迎制作更好的类，或者可以用来增强 (enhance) 这些类的包。因此，我们希望，当您考虑这样一个缺陷时，您的第一个想法是“我能做些什么来改进它?”

类似的考虑也适用于内核中实现设计决策的那些部分，其中许多应该留在类文件中，但不在当前系统中。我们意识到，在这种情况下，您自己纠正问题要困难得多，但在内核中进行这样的更改可能是我们的主要项目 (major project)；因此，此类增强必须等待 LaTeX3。

2 编写类和包

本节介绍与编写 LaTeX 类和包有关的一般要点 (general points)。

2.1 旧版本

如果要升级现有的 LaTeX 2.09 样式文件 (style file)，建议冻结 2.09 版本，不再维护它。20 世纪 90 年代早期存在各种 LaTeX 方言 (dialects)，为不同版本的 LaTeX 维护宏包几乎是不可能的。当然，对于某些组织来说，必须并行维护两个版本，但这对于个人支持的那些包和类并不是必需的：它们应仅支持新的标准 LaTeX 2_ε，而不是 LaTeX 的过时版本。

2.2 使用“docstrip”和“doc”

如果您打算为 LaTeX 编写一个大型的类或包，那么您应该考虑使用 LaTeX 附带的 `doc` 软件。使用 `doc` 软件编写的 LaTeX 类和包可以通过两种方式进行处理：它们可以通过 LaTeX 运行，以生成文档 (documentation)；并且可以使用 `docstrip` 处理它们，以生成 `.cls` 或 `.sty` 文件。

`doc` 软件可以自动生成定义的索引 (indexes of definitions)、命令使用的索引 (indexes of command use) 和更日志列表 (change-log lists)。它对于维护和记录大型 TeX 源 (sources) 非常有用。

LaTeX 内核本身和标准类 (standard classes) 等的文档来源 (documented sources) 是 `doc` 文档 (`doc` documents)；它们位于发行版 (distribution) 的 `.dtx` 文件中。实际上，您可以通过在 `source2e.tex` 上运行 LaTeX，将内核的源代码设置为一个长文档 (long document)，并完成索引。使用类文件 (class file) `ltxdoc.cls` 排版这些文档。

有关 `doc` 和 `docstrip` 的更多信息，请参阅 `docstrip.dtx`、`doc.dtx` 等文件和 *The LaTeX Companion, second edition*【《LaTeX 指南》，第二版】。有关它的使用示例，请查看 `.dtx` 文件。

2.3 它是一个类还是一个包？

当您想在文件中放入一些新的 LaTeX 命令时，首先要做的是决定它应该是*文档类* (*document class*) 还是*宏包* (*package*)。经验法则是：

如果这些命令可以与任何文档类 (document class) 一起使用，那么将其制作为一个包 (package)；如果不能，那么将它们制作为一个类 (class)。

有两种主要的类：像`article`(论文)、`report`(报告)、`letter`(信函) 等，它们是独立的；以及其他类的扩展 (extensions) 或变体 (variations)，例如 `proc` 文档类，它是在 `article` 文档类的基础上构建的。

因此，一家公司可能会有一个自有的 `ownlet` 类，用他们自己的标题信纸 (headed note-paper) 打印信函 (printing letters)。这样的类将构建在现有的 `letter` 类基础之上，但是它不能与任何其他文档类 (document class) 一起使用，所以我们使用 `ownlet.cls` 而不是 `ownlet.sty`。

相比之下，`graphics` 包提供了将图像包含到 LaTeX 文档中的命令。因为这些命令可以与任何文档类一起使用，所以我们使用了 `graphics.sty` 而不是 `graphics.cls`。

2.4 命令名

LaTeX 有三种类型的命令：

有一些是作者命令 (author commands)，例如 `\section`、`\emph` 和 `\times`：其中大多数都有短名称 (short names)，都是小写的 (lower case)。

还有类和包编写者命令 (class and package writer commands)：大多数命令都有长的混合大小写的名称，如下：

```
\InputIfFileExists  \RequirePackage  \PassOptionsToClass
```

最后，还有在 LaTeX 实现 (implementation) 中使用的内部命令 (internal commands)，例如 `\@tempcnta`、`\@ifnextchar` 和 `\@eha`：这些命令的名称中大多数包含 `@`，这意味着它们不能在文档 (documents) 中使用，只能在类和包文件中使用。

不幸的是，由于历史原因，这些命令之间的区别常常模糊不清。例如，`\hbox` 是一个内部命令，只能在 LaTeX 内核中使用，而 `\m@ne` 是常数 -1，也可能是 `\MinusOne`。

然而，这条经验法则 (rule of thumb) 仍然有用：如果命令的名称中有 `@`，那么它不是受支持的 LaTeX 语言的一部分—并且它的行为 (behaviour) 在未

来的版本 (future releases) 中可能会发生变化！如果一个命令是混合大小写的 (mixed-case)，或者是在 *LaTeX: A Document Preparation System*【《特可爱原本》】中描述了的，那么您可以依赖支持该命令的 LaTeX 2ε 的未来版本。

2.5 盒子命令和颜色

即使您不打算在自己的文档中使用颜色 (colour)，通过注意本节中的要点，也可以确保您的类 (class) 或包 (package) 与 `color` 包兼容。这可能让使用您的类或包的人受益，因为他们可以使用彩色打印机 (colour printers)。

确保“颜色安全 (colour safety)”的最简单方法是始终使用 LaTeX 盒子命令 (box commands) 而不是 TeX 原语 (primitives)，即使用 `\sbox` 而不是 `\setbox`，使用 `\mbox` 而不是 `\hbox` 和 `\parbox`，使用 `minipage` 环境而不是 `\vbox`。LaTeX 盒子命令有新的选项 (new options)，这意味着它们现在与 TeX 原语一样强大。

作为可能出错的示例，请考虑到在 `{\ttfamily` ⟨*text*⟩`}` 中，字体刚好在 `}` 之前恢复 (restored)，而在外观相似的结构 (similar looking construction) `{\color{green}`⟨*text*⟩`}` 中，颜色在最后一个 `}` 之后恢复。通常，这种区别根本无关紧要；但考虑为一个原始的 (primitive) TeX 盒赋值，例如：

```
\setbox0=\hbox{\color{green} ⟨text⟩}
```

现在颜色恢复 (colour-restore) 发生在 `}` 之后，因此不会存储在盒子中。这到底会产生什么样的不良影响取决于颜色的实现方式：它可能会在文档的其余部分获取错误的颜色，也可能会在用于打印文档的 dvi 驱动程序 (dvi-driver) 中导致错误。

同样有趣的是命令 `\normalcolor`。这通常只是 `\relax`(即什么也不做)，但您可以像使用 `\normalfont` 一样，将页面的区域 (regions of the page)(如标题或节标题) 设置为“主文档颜色 (main document colour)”。

2.6 定义文本和数学字符

由于 LaTeX 2ε 支持不同的编码 (encodings)，因此必须使用为此目的提供的命令和在 *LaTeX 2ε Font Selection*【《LaTeX 2ε 的字体选择》】中描述的命令，来定义用于生成符号 (symbols)、重音符号 (accents)、复合字形 (composite

glyphs) 等的命令。该系统的这一部分仍在开发中，因此应非常谨慎地执行此类任务。

此外，`\DeclareRobustCommand` 应该用于这种类型的独立于编码的命令 (encoding-independent commands)。

请注意，不再可能在数学模式 (math mode) 之外引用数学字体设置 (math font set-up)：例如，`\textfont 1` 和 `\scriptfont 2` 都不能在其他模式下定义。

2.7 常规样式

新系统提供了许多旨在帮助您生成结构良好的类和包文件的命令，这些文件既健壮 (robust) 又可移植 (portable)。本节概述了智能利用 (intelligent use) 这些资源的一些方法。

2.7.1 加载其他文件

LaTeX 提供以下命令：

新的描述
1995/12/01

```
\LoadClass          \LoadClassWithOptions
\RequirePackage     \RequirePackageWithOptions
```

用于在其他类或包中使用类或包。出于多种原因，我们强烈建议您使用它们，而不是原始的 (primitive) 的 `\input` 命令。

用 `\input` ⟨*filename*⟩ 加载的文件不会列在 `\listfiles` 列表中。

如果一个包总是用 `\RequirePackage...` 或 `\usepackage` 加载，那么即使多次请求加载，也只会加载一次。相反，如果用 `\input` 加载，则可以多次加载；这样的额外加载可能会浪费时间和内存，并且可能会产生奇怪的结果。

如果一个包提供了选项处理 (option-processing)，则同样，如果包是 `\input` 而不是通过 `\usepackage` 或 `\RequirePackage...` 加载的，则可能会出现奇怪的结果。

如果 `foo.sty` 包通过使用 `\input baz.sty` 加载 `baz.sty` 包，则用户会得到一条警告：

```
LaTeX Warning: You have requested package `foo',
               but the package provides `baz'.
```

```
警告：您已请求包“foo”，
      但该包提供了“baz”。
```

因此，出于几个原因，使用 \input 加载包不是一个好主意。

不幸的是，如果要将文件 myclass.sty 升级为类文件 (class file)，那么必须确保包含 \input myclass.sty 的任何旧文件仍然可以工作。

对于标准类 (article、book 和 report) 也是如此，因为许多现有的 LaTeX 2.09 文档样式 (document styles) 都包含 \input article.sty。我们用来解决这个问题的方法是提供最少的文件 article.sty、book.sty 和 report.sty，它们只是加载适当的类文件 (class files)。

例如，article.sty 只包含以下几行:

```
\NeedsTeXFormat{LaTeX2e}
\@obsoletefile{article.cls}{article.sty}
\LoadClass{article}
```

您可能希望这样做，或者，如果您认为这样做是安全的，您可以决定删除 myclass.sty。

2.7.2 让它健壮

我们认为在编写包和类时，尽可能多地使用 LaTeX 命令是一种很好的做法 (practice)。

因此，我们建议不要使用 \def...，而应使用 \newcommand、\renewcommand 或 \providecommand 中的一个；\CheckCommand 也很有用。这样做可以减少您无意中重新定义命令，从而产生意外结果的可能性。

定义环境 (environment) 时，请使用 \newenvironment 或 \renewenvironment 代替 \def\foo{...} 或 \def\endfoo{...}。

如果需要设置 (set) 或更改 ⟨*dimen*⟩ 或 ⟨*skip*⟩ 寄存器 (register) 的值，请使用 \setlength。

要操作盒子 (boxes)，请使用诸如 \sbox、\mbox 或 \parbox 等 LaTeX 命令，而不是 \setbox、\hbox 和 \vbox。

使用 \PackageError、\PackageWarning 或 \PackageInfo (或等效的类命令)，而不是 \@latexerr、\@warning 或 \wlog。

仍然可以通过定义 \ds@⟨*option*⟩ 和调用 \@options 来声明选项 (declare options)；但是我们推荐使用 \DeclareOption 和 \ProcessOptions 命令。它们更加强大，并且使用更少的内存。因此，不要使用：

```
\def\ds@draft{\overfullrule 5pt}
\@options
```

您应该使用：

```
\DeclareOption{draft}{\setlength{\overfullrule}{5pt}}
\ProcessOptions\relax
```

这种做法的优点是，您的代码更具可读性，更重要的是，当与未来版本的 LaTeX 一起使用时，它不太可能被破坏。

2.7.3 使它可移植

让你的文件尽可能的可移植 (portable) 也是明智的。确保这一点；它们应该只包含可见的 7 位文本 (visible 7-bit text)；文件名 (filenames) 最多应包含八个字符 (characters)(加上三个字母的扩展名)。当然，它不能与标准 LaTeX 发行版中的文件同名，但其内容可能与其中一个文件相似。

如果本地 (local) 类或包有一个通用前缀 (common prefix)，这也很有用，例如 Nowhere 大学的类 (classes) 可能以 unw 开头。这有助于避免每所大学将自己的论文类 (thesis class) 都称为 thesis.cls。

如果您依赖于 LaTeX 内核的某些特性 (features)，或者依赖于一个包，请指定您需要的发布日期 (release-date)。例如，包错误命令 (package error commands) 是在 1994 年 6 月的版本中引入的，因此，如果您使用它们，则应输入：

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

2.7.4 有用的钩子

一些宏包 (packages) 和文档样式 (document styles) 必须重新定义命令 \document 或 \enddocument 才能实现其目标 (goal)。但现在这不再是必要的。

现在您可以使用“钩子 (hooks)”`\AtBeginDocument` 和 `\AtEndDocument` (参见第 4.6 节)。同样，使用这些钩子可以减少您的代码被未来版本 (future versions) 的 LaTeX 破坏的可能性。它还使您的宏包更有可能与其他人的宏包协同工作。

但是，请注意，`\AtBeginDocument` 钩子中的代码是前言的一部分。因此，对可以放在那里的东西有限制；特别是，不能进行排版 (typesetting)。

新的描述
1996/12/01

3 类或包的结构

LaTeX 2$_\varepsilon$ 类和包的结构比 LaTeX 2.09 样式文件 (style files) 的结构更多。类或包文件的轮廓 (outline) 如下:

标识符 (Identification) 该文件说明它是一个 LaTeX 2$_\varepsilon$ 包或类，并对其本身进行了简短描述。

初步声明 (Preliminary declarations) 在这里，文件声明了一些命令，还可以加载其他文件。通常，这些命令只是声明选项中使用的代码所需的命令。

选项 (Options) 该文件声明并处理其选项。

更多的声明 (More declarations) 这是文件完成大部分工作的地方：声明新的变量、命令和字体，以及加载其他文件。

3.1 标识符

类或包文件所做的第一件事是标识自身 (identify itself)。包文件 (package files) 是这样做的：

```
\NeedsTeXFormat{LaTeX2e}
\ProvidesPackage{⟨package⟩}[⟨date⟩ ⟨other information⟩]
```

举例：

```
\NeedsTeXFormat{LaTeX2e}
\ProvidesPackage{latexsym}[1994/06/01 Standard LaTeX package]
```

类文件 (class files) 是这样做的：

```
\NeedsTeXFormat{LaTeX2e}
\ProvidesClass{⟨class-name⟩}[⟨date⟩ ⟨other information⟩]
```

举例：

```
\NeedsTeXFormat{LaTeX2e}
\ProvidesClass{article}[1994/06/01 Standard LaTeX class]
```

这个 ⟨*date*⟩ 应以“YYYY/MM/DD”的形式给出，如果使用可选参数，则它必须存在 (对于 `\NeedsTeXFormat` 命令也是如此)。此语法的任何派生 (derivation) 都将导致低级别的 (low-level) TEX 错误—命令期望有效的语法来加快包或类的日常使用 (daily usage)，并且没有为开发人员犯错的情况做任何准备！

新的描述
1998/06/19

每当用户在其 `\documentclass` 或 `\usepackage` 命令中指定日期时，都会选中此日期。例如，如果您写道：

```
\documentclass{article}[1995/12/23]
```

然后，位于不同位置 (location) 的用户将收到一条警告，指出他们的 `article` 副本已过期。

使用类 (class) 时会显示类的描述。包 (package) 的描述放入日志文件 (log file) 中。这些描述也由 `\listfiles` 命令显示。除标准 LATEX 发行版中的文件外，不得在任何文件的身份识别标志 (identification banner) 中使用短语 `Standard LaTeX`

3.2 使用类和包

LATEX 2.09 样式文件 (style files) 与 LATEX 2$_\varepsilon$ 包和类之间的第一个主要区别是，LATEX 2$_\varepsilon$ 支持模块化 (*modularity*)，意思是从小的构建块 (small building-blocks) 构建文件，而不是使用大的单个文件 (large single files)。

按如下的方法加载 LATEX 包和类：

```
\RequirePackage[⟨options⟩]{⟨package⟩}[⟨date⟩]
```

举例：

```
\RequirePackage{ifthen}[1994/06/01]
```

该命令与作者的命令 `\usepackage` 具有相同的语法。它允许包或类使用其他包提供的特性 (features)。例如，通过加载 `ifthen` 包，包编写者可以使用该包提供的“if...then...else...”命令。

LATEX 类可以加载另一个类，如下所示：

```
\LoadClass[⟨options⟩]{⟨class-name⟩}[⟨date⟩]
```

举例：

```
\LoadClass[twocolumn]{article}
```

此命令的语法与作者的命令 `\documentclass` 相同。它允许类基于另一个类的语法 (syntax) 和外观 (appearance)。例如，通过加载 `article` 类，类编写者只需更改 `article` 中他们不喜欢的那一部分，而不必从头开始编写新类 (new class)。

下面的命令可以在通常的情况下 (common case) 使用，即您只需加载一个类或包文件，其中包含当前类正在使用的选项。 新的特性 1995/12/01

```
\LoadClassWithOptions{⟨class-name⟩}[⟨date⟩]
\RequirePackageWithOptions{⟨package⟩}[⟨date⟩]
```

举例：

```
\LoadClassWithOptions{article}
\RequirePackageWithOptions{graphics}[1995/12/01]
```

3.3 声明选项

LaTeX 2.09 样式与 LaTeX 2_ε 包和类之间的另一个主要区别是选项处理 (option handling)。包和类现在可以声明选项 (declare options)，这些选项可以由作者指定；例如，`article` 类声明 `twocolumn` 选项。注意，一个选项的名称应该只包含那些在“LaTeX 名称”中允许的字符 (characters)，特别是它不能包含任何控制序列 (control sequences)。 新的描述 1998/12/01

选项的声明如下：

```
\DeclareOption{⟨option⟩}{⟨code⟩}
```

例如，`graphics` 包的 `dvips` 选项 (略加简化) 的实现如下：

```
\DeclareOption{dvips}{\input{dvips.def}}
```

这意味着当作者编写 `\usepackage[dvips]{graphics}` 时，文件 `dvips.def` 将被加载。另一个例子是，在 `article` 类中声明了 `a4paper` 选项来设置 `\paperheight` 和 `\paperwidth`：

```
\DeclareOption{a4paper}{%
   \setlength{\paperheight}{297mm}%
   \setlength{\paperwidth}{210mm}%
}
```

有时，用户会请求类或包未显式声明的 (not explicitly declared) 选项。默认情况下，这将产生警告 (针对类) 或错误 (针对包)，这种行为可以改变 (behaviour) 如下：

```
\DeclareOption*{⟨code⟩}
```

例如，要使该 `fred` 包对未知选项 (unknown options) 产生警告 (warning) 而不是错误 (error)，您可以指定：

```
\DeclareOption*{%
   \PackageWarning{fred}{Unknown option `\CurrentOption'}%
}
```

然后,如果作者编写了 `\usepackage[foo]{fred}`,他们将得到一个警告 `Package fred Warning: Unknown option `foo'.`。另一个例子是，每当使用 ⟨*ENC*⟩ 选项时，`fontenc` 包尝试加载一个文件 ⟨*ENC*⟩`enc.def`，这可以通过以下方式实现：

```
\DeclareOption*{%
   \input{\CurrentOption enc.def}%
}
```

可以使用命令 `\PassOptionsToPackage` 或 `\PassOptionsToClass` 将选项传递到另一个包或类 (请注意，这是一个仅适用于选项名称的专业操作)。例如，要将每个未知选项 (unknown option) 传递给 `article` 类，您可以使用：

新的描述
1998/12/01

```
\DeclareOption*{%
   \PassOptionsToClass{\CurrentOption}{article}%
}
```

如果这样做，那么您应该确保在以后的某个时候加载该类，否则这些选项将永远不会被处理！

到目前为止，我们只解释了如何声明 (declare) 选项，而不是如何执行 (execute) 它们。要处理调用文件时使用的选项，应该使用：

```
\ProcessOptions\relax
```

这将对指定和声明的每个选项执行 ⟨*code*⟩ (请参阅第 4.7 节了解如何执行此操作的详细信息)。

例如，如果 `jane` 宏包文件包含：

```
\DeclareOption{foo}{\typeout{Saw foo.}}
\DeclareOption{baz}{\typeout{Saw baz.}}
\DeclareOption*{\typeout{What's \CurrentOption?}}
\ProcessOptions\relax
```

然后一个作者写了 `\usepackage[foo,bar]{jane}`，然后他们就会看到这些消息：`Saw foo.` and `What's bar?`

3.4 最小的类文件

类或包的大部分工作是定义新命令 (defining new commands) 或更改文档的外观 (appearance)。这是在包的主体 (body of the package) 中完成的，可以使用 `\newcommand` 或 `\setlength` 等命令。

LaTeX 2ε 提供了几个新命令来帮助类和包作者，这些命令在第 4 节中有详细描述。

每个类文件 (class file) 必须包含四个内容:`\normalsize` 的定义、`\textwidth` 的值、`\textheight` 的值、页码 (page-numbering) 的规范 (specification)。因此，最小文档类 (minimal document class file) 文件[1]如下所示：

```
\NeedsTeXFormat{LaTeX2e}
\ProvidesClass{minimal}[1995/10/30 Standard LaTeX minimal class]
\renewcommand{\normalsize}{\fontsize{10pt}{12pt}\selectfont}
\setlength{\textwidth}{6.5in}
\setlength{\textheight}{8in}
\pagenumbering{arabic}       % 即使此类不会显示页码，也需要
```

然而，这个类文件 (class file) 不支持脚注 (footnotes)、边距 (marginals)、浮动 (floats) 等，也不提供任何 2 个字母的字体命令 (2-letter font commands)，如`\rm`；因此，大多数类将包含超过此最小值的内容！

[1]这个类现在在标准发行版中，是 `minimal.cls`。

3.5 举例：一个本地信函类

公司可能有自己的信函类 (letter class)，用于按公司样式设置字母。本节显示了这样一个类的简单实现 (implementation)，尽管真正的类需要更多的结构 (structure)。

该类首先宣布自己为 `neplet.cls`。

```
\NeedsTeXFormat{LaTeX2e}
\ProvidesClass{neplet}[1995/04/01 NonExistent Press letter class]
```

然后，下一段将所有选项传递给 `letter` 类，`letter` 类加载了 `a4paper` 选项。

```
\DeclareOption*{\PassOptionsToClass{\CurrentOption}{letter}}
\ProcessOptions\relax
\LoadClass[a4paper]{letter}
```

为了使用公司信头 (letter head)，它重新定义了 `firstpage` 页面样式 (page style)：`firstpage` 是信函第一页上使用的页面样式。

```
\renewcommand{\ps@firstpage}{%
   \renewcommand{\@oddhead}{⟨letterhead goes here⟩}%
   \renewcommand{\@oddfoot}{⟨letterfoot goes here⟩}%
}
```

就是这样！

3.6 举例：一个简报类

一份简单的简报 (newsletter) 可以通过 LaTeX 用 `article` 类的变体来排版。该变体首先将自己声明为 `smplnews.cls`。

```
\NeedsTeXFormat{LaTeX2e}
\ProvidesClass{smplnews}[1995/04/01 The Simple News newsletter class]

\newcommand{\headlinecolor}{\normalcolor}
```

它将大多数指定的选项传递给 `article` 类：除了关闭的 (switched off) `onecolumn` 选项和将标题 (headline) 设置为绿色的 `green` 选项之外。

```
\DeclareOption{onecolumn}{\OptionNotUsed}
\DeclareOption{green}{\renewcommand{\headlinecolor}{\color{green}}}

\DeclareOption*{\PassOptionsToClass{\CurrentOption}{article}}

\ProcessOptions\relax
```

然后它加载带有 `twocolumn` 选项的 `article` 类：

```
\LoadClass[twocolumn]{article}
```

由于简报 (newsletter) 将彩色打印，现在它将加载 `color` 宏包。该类没有指定设备驱动程序选项 (device driver option)，因为这应该由 `smplnews` 类的用户指定。

```
\RequirePackage{color}
```

然后，该类重新定义 `\maketitle`，以使用合适的颜色以 72pt Helvetica 粗斜体生成标题 (title)。

```
\renewcommand{\maketitle}{%
   \twocolumn[%
      \fontsize{72}{80}\fontfamily{phv}\fontseries{b}%
      \fontshape{sl}\selectfont\headlinecolor
      \@title
   ]%
}
```

它重新定义 `\section` 并关闭节编号 (section numbering)。

```
\renewcommand{\section}{%
   \@startsection
      {section}{1}{0pt}{-1.5ex plus -1ex minus -.2ex}%
      {1ex plus .2ex}{\large\sffamily\slshape\headlinecolor}%
}

\setcounter{secnumdepth}{0}
```

它还设置了三个基本要素 (essential things)：

```
\renewcommand{\normalsize}{\fontsize{9}{10}\selectfont}
\setlength{\textwidth}{17.5cm}
\setlength{\textheight}{25cm}
```

实际上，一个类需要的不仅仅是这些：它还为期刊编号 (issue numbers)、文章作者、页面样式 (page styles) 等提供命令，这个框架只是一个开始。`ltnews` 类文件并不比这个文件复杂多少。

4 用于类和包作者的命令

本节简要介绍用于类和包作者的每个新命令。要了解这个新系统的其他方面，您还应该阅读：

LaTeX: A Document Preparation System【《LaTeX：一个文档准备系统》】

The LaTeX Companion, second edition 【《LaTeX 指南》，第二版】

LaTeX 2ε for Authors 【《LaTeX 2ε 作者指南》】

4.1 标识符

这里讨论的第一组命令是用于标识 (identify) 类或包文件的命令。

`\NeedsTeXFormat {`⟨*format-name*⟩`} [`⟨*release-date*⟩`]`

此命令告诉 TeX，应使用具有 ⟨*format-name*⟩ 名称的格式处理此文件。您可以使用可选参数 ⟨*release-date*⟩ 以进一步指定所需格式的最早发布日期 (earliest release date)。当格式的发布日期早于指定日期时，将发出警告。标准的 ⟨*format-name*⟩ 是 `LaTeX2e`。日期 (如果有) 的格式必须为 YYYY/MM/DD。

举例：

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

`\ProvidesClass {`⟨*class-name*⟩`} [`⟨*release-info*⟩`]`
`\ProvidesPackage {`⟨*package-name*⟩`} [`⟨*release-info*⟩`]`

这声明当前文件 (current file) 包含文档类 ⟨*class-name*⟩ 或宏包 ⟨*package-name*⟩ 的定义

可选的 ⟨*release-info*⟩，如果使用，必须包含：

- 此版本文件的发布日期，格式为 YYYY/MM/DD；
- 可以选择后跟一个空格和一个简短的描述，可能包括版本号 (version number)。

必须严格遵循上面的语法，这样 `\LoadClass` 或 `\documentclass`(对于类)，`\RequirePackage` 或 `\usepackage` (对于包) 才能使用这些信息来测试发行版 (release) 是否太旧。

所有这些 ⟨*release-info*⟩ 信息都由 `\listfiles` 显示，因此不应该太长。

举例：

```
\ProvidesClass{article}[1994/06/01 v1.0 Standard LaTeX class]
\ProvidesPackage{ifthen}[1994/06/01 v1.0 Standard LaTeX package]
```

`\ProvidesFile {`⟨*file-name*⟩`} [`⟨*release-info*⟩`]`

这与前两个命令类似，只是这里必须给出完整的文件名，包括扩展名。它用于声明除主类 (main class) 和包 (package) 文件以外的任何文件

举例：

```
\ProvidesFile{T1enc.def}[1994/06/01 v1.0 Standard LaTeX file]
```

请注意，除了标准 LaTeX 发行版中的文件外，不得在任何文件的身份识别标志 (identification banner) 中使用短语 `Standard LaTeX`。

4.2 加载文件

这组命令可用于通过构建现有类或包来创建自己的文档类 (document class) 或包。

新的特性 1995/12/01

`\RequirePackage [`⟨*options-list*⟩`] {`⟨*package-name*⟩`} [`⟨*release-info*⟩`]`
`\RequirePackageWithOptions {`⟨*package-name*⟩`} [`⟨*release-info*⟩`]`

包和类应该使用这些命令来加载其他包。

`\RequirePackage` 的使用与作者命令 (author command) `\usepackage` 相同。

举例：

```
\RequirePackage{ifthen}[1994/06/01]
\RequirePackageWithOptions{graphics}[1995/12/01]
```

`\LoadClass` [⟨*options-list*⟩] {⟨*class-name*⟩} [⟨*release-info*⟩]
`\LoadClassWithOptions` {⟨*class-name*⟩} [⟨*release-info*⟩]

这些命令仅用于类文件 (class files)，不能用于包文件 (packages files)；它们在类文件中最多只能使用一次。 新的特性 1995/12/01

使用 `\LoadClass` 与使用 `\documentclass` 加载类文件相同。

举例：

```
\LoadClass{article}[1994/06/01]
\LoadClassWithOptions{article}[1995/12/01]
```

两个 `WithOptions` 版本只需加载类 (或包) 文件，其中包含当前文件 (类或包) 正在使用的选项。关于其使用的进一步讨论，见下文第 4.5 节。 新的特性 1995/12/01

4.3 选项声明

下面的命令处理文档类及包的选项声明 (declaration) 和选项使用 (handling)。每个选项的名称必须是“LaTeX 名称 (LaTeX name)”。 新的描述 1998/12/01

有一些命令是专门为在这些命令 (见下文) 的 ⟨*code*⟩ 参数中使用而设计的。

`\DeclareOption` {⟨*option-name*⟩} {⟨*code*⟩}

这使得 ⟨*option-name*⟩ 成为放置它的类或包的“声明选项 (declared option)”。

如果为类或软件包指定该选项，则 ⟨*code*⟩ 参数包含要执行的代码；它可以包含任何有效的 LaTeX 2_ε 构造 (LaTeX 2_ε construct)。

举例：

```
\DeclareOption{twoside}{\@twosidetrue}
```

`\DeclareOption*` {⟨*code*⟩}

这声明了要为指定但没有明确声明的类或包的每个选项执行 ⟨*code*⟩，此代码称为“默认选项代码 (default option code)”，它可以包含任何有效的 LaTeX 2_ε 构造 (LaTeX 2_ε construct)。

如果类文件不包含 \DeclareOption*，则默认情况下，该类的所有指定的 (specified) 但未声明的 (undeclared) 选项都将自动传递给所有包 (该类的指定的和声明的选项也一样)。

如果包文件不包含 \DeclareOption*，则默认情况下，包的每个指定的但未声明的选项都将产生错误。

4.4 选项代码中的命令

这两个命令只能在 \DeclareOption 或 \DeclareOption* 的 ⟨*code*⟩ 参数中使用。这些参数中常用的其他命令可以在接下来的几个小节中找到。

\CurrentOption

这将扩展到当前选项的名称。

\OptionNotUsed

这会将当前选项 (current option) 添加到“未使用的选项 (unused options)”列表中。

您现在可以在这些 ⟨*code*⟩ 参数中包含散列标记 (hash marks)(#)，而不需要进行特殊处理 (以前，需要将它们翻倍)。 新的特性 1995/06/01

4.5 四处移动选项

这两个命令在 \DeclareOption 或 \DeclareOption* 的 ⟨*code*⟩ 参数中也非常有用：

\PassOptionsToPackage {⟨*options-list*⟩} {⟨*package-name*⟩}
\PassOptionsToClass {⟨*options-list*⟩} {⟨*class-name*⟩}

这意味着它将 ⟨*option-list*⟩ 添加到未来 \RequirePackage 或 \usepackage 命令使用的选项列表中，\RequirePackage 或 \usepackage 命令用于 ⟨*package-name*⟩。

举例：

```
\PassOptionsToPackage{foo,bar}{fred}
\RequirePackage[baz]{fred}
```

和下面的命令是一样的：

```
\RequirePackage[foo,bar,baz]{fred}
```

类似地，类文件中可以使用 \PassOptionsToClass 将选项传递给要用 \LoadClass 加载的另一个类。

这两个命令的效果 (effects) 和用法 (use) 应与以下两个命令 (见上文第 4.2 节) 的效果和用法进行对比：

新的描述
1995/12/01

```
\LoadClassWithOptions
\RequirePackageWithOptions
```

命令 \RequirePackageWithOptions 与 \RequirePackage 类似，但它总是使用与当前类或包使用的选项列表完全相同的选项列表加载所需的包，而不是使用 \PassOptionsToPackage 显式提供或传递的任何选项。

\LoadClassWithOptions 的主要用途是允许一个类简单地构建在另一个类上，例如：

```
\LoadClassWithOptions{article}
```

这应该与下面的略有不同的结构进行比较：

```
\DeclareOption*{\PassOptionsToClass{\CurrentOption}{article}}
\ProcessOptions\relax
\LoadClass{article}
```

如上所述,效果大致相同,但第一种要键入的内容要少得多,而且 \LoadClassWithOptions 方法运行得稍微快一些。

然而，如果类声明了自己的选项，那么这两种结构是不同的。比较下面两个示例：
第一个示例：

```
\DeclareOption{landscape}{\@landscapetrue}
\ProcessOptions\relax
\LoadClassWithOptions{article}
```

第二个示例：

```
\DeclareOption{landscape}{\@landscapetrue}
\DeclareOption*{\PassOptionsToClass{\CurrentOption}{article}}
\ProcessOptions\relax
\LoadClass{article}
```

在第一个示例中，当使用此选项调用当前类 (current class) 时，将精确使用 `landscape` 选项加载 article 类。相比之下，在第二个示例中，它永远不会用 `landscape` 选项调用，因为在这种情况下，article 仅由默认选项处理程序 (default option handler) 传递选项，但此处理程序 (handler) 不用于 `landscape`，因为该选项是显式声明的。

4.6 延迟代码

前两个命令也主要用于 `\DeclareOption` 或 `\DeclareOption*` 的 ⟨*code*⟩ 参数中。

`\AtEndOfClass {`⟨*code*⟩`}`
`\AtEndOfPackage {`⟨*code*⟩`}`

这些命令声明 ⟨*code*⟩，⟨*code*⟩ 在内部保存起来，然后在处理完当前类或包文件之后执行。

允许重复使用这些命令：参数中的代码按其声明的顺序存储 (并在以后执行)。

`\AtBeginDocument {`⟨*code*⟩`}`
`\AtEndDocument {`⟨*code*⟩`}`

这些命令声明 ⟨*code*⟩ 要在内部保存并在 LaTeX 执行 `\begin{document}` 或 `\end{document}` 时执行。

在设置字体选择表 (font selection tables) 之后，在 `\begin{document}` 代码末尾附近执行 `\AtBeginDocument` 的参数中指定的 ⟨*code*⟩。因此，在为排版做好一切准备之后，当文档的正常字体 (normal font for the document) 为当前字体 (current font) 时，它是放置需要执行的代码的一个有用位置。

`\AtBeginDocument` 钩子 (hook) 不能用于执行任何排版的代码，因为排版 新的描述 1995/12/01

结果是不可预测的。

在最后页面 (final page) 完成之前，在处理完任何剩余的浮动环境 (floating environments) 之前，在 `\end{document}` 代码开头执行 `\AtEndDocument` 的参数中指定的 ⟨*code*⟩。如果要在这两个过程 (processes) 之后执行某些 ⟨*code*⟩，则应在 ⟨*code*⟩ 中的适当位置中包含一个 `\clearpage`。

允许重复使用这些命令：参数中的代码按其声明的顺序存储 (并在以后执行)。

`\AtBeginDvi {`⟨*specials*⟩`}`

新的特性
1994/12/01

这些命令将保存在盒子寄存器 (box register) 中，在文档第一页的“shipout”开始处，将盒子寄存器的内容写入 `.dvi` 文件。

这不能用于将任何排版材料添 (typeset material) 加到 `.dvi` 文件中。

允许重复使用此命令。

4.7 选项处理

`\ProcessOptions`

此命令对每个选定的选项执行 ⟨*code*⟩。

我们将首先描述 `\ProcessOptions` 如何在包文件 (package file) 中工作，然后描述它在类文件 (class file) 中的不同之处。

要详细了解 `\ProcessOptions` 在包文件中的作用，您必须了解*局部* (*local*) 选项和*全局* (*global*) 选项之间的区别。

- **局部选项** (**Local options**) 是在其中任何一个的 ⟨*options*⟩ 参数中为这个特定包明确指定的选项：

  `\PassOptionsToPackage{`⟨*options*⟩`}` `\usepackage[`⟨*options*⟩`]`
  `\RequirePackage[`⟨*options*⟩`]`

- **全局选项** (**Global options**) 是作者在 `\documentclass[`⟨*options*⟩`]` 的 ⟨*options*⟩ 参数中指定的任何其他选项。

例如，假设一个文档开始于：

```
\documentclass[german,twocolumn]{article}
\usepackage{gerhardt}
```

同时 gerhardt 包调用 fred 包和：

```
\PassOptionsToPackage{german,dvips,a4paper}{fred}
\RequirePackage[errorshow]{fred}
```

然后：

- fred 的局部选项 (local options) 是german、dvips, a4paper、errorshow；
- fred 唯一的全局选项 (global option) 是twocolumn。

调用 \ProcessOptions 时，会发生以下情况：

- 首先，对于迄今为止通过 \DeclareOption 在 fred.sty 中声明的每个选项，可以查看该选项是 fred 的全局选项还是局部选项：如果是，则执行相应的代码。

  这是按照在 fred.sty 中声明这些选项的顺序进行的。

- 然后，对于其余的每个局部选项，\ds@⟨*option*⟩ 命令如果已在某处定义过 (而不是通过 \DeclareOption)，则执行该命令；否则，将执行“默认选项代码 (default option code)”。如果未声明默认选项代码，则会生成错误消息。

  这是按照指定这些选项的顺序进行的。

在整个过程中，系统确保为选项声明的代码最多执行一次。

返回到示例，如果 fred.sty 包含：

```
\DeclareOption{dvips}{\typeout{DVIPS}}
\DeclareOption{german}{\typeout{GERMAN}}
\DeclareOption{french}{\typeout{FRENCH}}
\DeclareOption*{\PackageWarning{fred}{Unknown `\CurrentOption'}}
\ProcessOptions\relax
```

那么处理此文档的结果将是：

```
DVIPS
GERMAN
Package fred Warning: Unknown `a4paper'.
Package fred Warning: Unknown `errorshow'.
```

请注意以下事项：

- `dvips` 选项的代码在 `german` 选项的代码之前执行，因为这是它们在 `fred.sty` 中声明的顺序；
- 当处理声明的选项时，`german` 选项的代码只执行一次；
- `a4paper` 和 `errorshow` 选项会按 `\DeclareOption*` 声明的代码生成警告 (按指定的顺序)，而 `twocolumn` 选项不会：这是因为 `twocolumn` 是一个全局选项。

在类文件中，`\ProcessOptions` 的工作方式是相同的，只不过：所有选项都是局部的；`\DeclareOption*` 的默认值是 `\OptionNotUsed` 而不是错误。

请注意，因为 `\ProcessOptions` 有一个 `*` 格式，所以明智的做法是像前面的示例一样，在非星形格式后面加上 `\relax`，因为这样可以防止发出不必要的前瞻性错误消息和可能产生误导的错误消息。

新的描述 1995/12/01

`\ProcessOptions*`
`\@options`

这与 `\ProcessOptions` 类似，但它以调用命令中指定的顺序执行选项，而不是以类或包中声明的顺序执行。对于包，这意味着首先处理全局选项 (global options)。

为了简化将旧文档样式 (old document styles) 更新为 LaTeX 2$_\varepsilon$ 类文件的任务，LaTeX 2.09 中的 `\@options` 命令与之等效。

`\ExecuteOptions {`⟨*options-list*⟩`}`

对于 ⟨*options-list*⟩ 中的每个选项，按照顺序，这个命令只执行 `\ds@`⟨*option*⟩ 命令 (如果未定义此命令，则会自动忽略该选项)。

它可用于在 `\ProcessOptions` 之前提供“默认选项列表 (default option list)”。例如，假设在类文件中，您希望将默认设计 (default design) 设置为：双面打印，11pt 字体，分为两列。然后可以指定：

```
\ExecuteOptions{11pt,twoside,twocolumn}
```

4.8 安全文件命令

这些命令处理文件输入 (file input)，它们确保可以以用户友好的方式处理不存在的请求文件 (requested file)。

`\IfFileExists {`⟨*file-name*⟩`} {`⟨*true*⟩`} {`⟨*false*⟩`}`

如果文件存在，则执行 ⟨*true*⟩ 中指定的代码。

如果文件不存在，则执行 ⟨*false*⟩ 中指定的代码。

此命令不输入文件。

`\InputIfFileExists {`⟨*file-name*⟩`} {`⟨*true*⟩`} {`⟨*false*⟩`}`

如果存在文件 ⟨*file-name*⟩，则输入该文件，并且在输入之前立即执行在 ⟨*true*⟩ 中指定的代码。

如果文件不存在，则执行 ⟨*false*⟩ 中指定的代码。

它是使用 `\IfFileExists` 实现的。

4.9 报告错误等

第三方类和包应该使用这些命令来报告错误，或者向作者提供信息。

`\ClassError {`⟨*class-name*⟩`} {`⟨*error-text*⟩`} {`⟨*help-text*⟩`}`
`\PackageError {`⟨*package-name*⟩`} {`⟨*error-text*⟩`} {`⟨*help-text*⟩`}`

这些将生成错误消息 (error message)。显示这个 ⟨*error-text*⟩ 和错误提示 (error prompt)。如果用户键入 `h`，则会显示 ⟨*help-text*⟩。

在 ⟨*error-text*⟩ 和 ⟨*help-text*⟩ 中:`\protect` 可用于阻止命令展开;`\MessageBreak` 会导致换行 (line-break)；`\space` 会打印空格 (space)。

请注意 ⟨*error-text*⟩ 将添加一个句号 (full stop)，因此不要在参数中添加句号。

举例：

```
\newcommand{\foo}{FOO}
\PackageError{ethel}{%
   Your hovercraft is full of eels,\MessageBreak
   and \protect\foo\space is \foo
}{%
   Oh dear! Something's gone wrong.\MessageBreak
   \space \space Try typing \space <return>
   \space to proceed, ignoring \protect\foo.
}
```

结果显示如下：

```
! Package ethel Error: Your hovercraft is full of eels,
(ethel)                and \foo is FOO.

See the ethel package documentation for explanation.
```

如果用户键入 `h`，将显示：

```
Oh dear! Something's gone wrong.
  Try typing  <return>  to proceed, ignoring \foo.
```

`\ClassWarning` {⟨*class-name*⟩} {⟨*warning-text*⟩}
`\PackageWarning` {⟨*package-name*⟩} {⟨*warning-text*⟩}
`\ClassWarningNoLine` {⟨*class-name*⟩} {⟨*warning-text*⟩}
`\PackageWarningNoLine` {⟨*package-name*⟩} {⟨*warning-text*⟩}
`\ClassInfo` {⟨*class-name*⟩} {⟨*info-text*⟩}
`\PackageInfo` {⟨*package-name*⟩} {⟨*info-text*⟩}

这四个 `Warning` 命令与错误命令 (error commands) 类似，只是它们只在屏幕上生成警告 (warning)，没有错误提示 (error prompt)。

前两个 `Warning` 版本也显示发生警告的行号 (line number)，而后两个 `WarningNoLine` 版本则不显示行号。

这两个 `Info` 命令非常相似，只是它们只将信息记录在记录文件 (transcript file) 中，包括行号 (line number)。这两者没有 `NoLine` 版本。

在 ⟨*warning-text*⟩ 和 ⟨*info-text*⟩ 中，`\protect` 可用于阻止命令展开 (expanding)；`\MessageBreak` 会导致换行 (line-break)；`\space` 会打印一个空格 (space)。此外，这些不应以句号 (full stop) 结束，因为会自动添加一个句号。

4.10 定义命令

LaTeX 2$_\varepsilon$ 提供了一些额外的方法来 (重新) 定义用于类和包文件中的命令。

这些命令的 * 形式应用于定义用 TeX 术语来说不是很长的命令。对于参数不打算包含整段文本的命令，这对于错误捕获 (error-trapping) 非常有用。 新的特性 1994/12/01

`\DeclareRobustCommand` {⟨*cmd*⟩} [⟨*num*⟩] [⟨*default*⟩] {⟨*definition*⟩}
`\DeclareRobustCommand*` {⟨*cmd*⟩} [⟨*num*⟩] [⟨*default*⟩] {⟨*definition*⟩}

此命令采用与 `\newcommand` 相同的参数，但它声明了一个健壮的命令 (robust command)，即使 ⟨*definition*⟩ 是脆弱的 (fragile)。您可以使用此命令定义新的健壮命令，或重新定义现有命令并使其健壮。如果重新定义了命令，则会将日志 (log) 放入记录文件 (transcript file) 中。

例如，如果 `\seq` 定义如下：

```
\DeclareRobustCommand{\seq}[2][n]{%
  \ifmmode
    #1_{1}\ldots#1_{#2}%
  \else
    \PackageWarning{fred}{You can't use \protect\seq\space in text}%
  \fi
}
```

然后，可以在移动参数 (moving arguments) 时使用命令 `\seq`，即使 `\ifmmode` 不能，例如：

```
\section{Stuff about sequences $\seq{x}$}
```

还请注意，在定义的开头，不需要在 `\ifmmode` 之前放一个 `\relax`；这是因为，这是因为这种 `\relax` 所提供的防止在错误的时间进行展开的保护 (protection) 将在内部提供。

`\CheckCommand` {⟨*cmd*⟩} [⟨*num*⟩] [⟨*default*⟩] {⟨*definition*⟩}
`\CheckCommand*` {⟨*cmd*⟩} [⟨*num*⟩] [⟨*default*⟩] {⟨*definition*⟩}

它使用与 `\newcommand` 相同的参数，但不是定义 ⟨*cmd*⟩，而是检查 ⟨*cmd*⟩ 的当前定义是否与 ⟨*definition*⟩ 给出的定义完全一致。如果这些定义不同，则会引发错误。

此命令对于在包开始更改命令定义之前检查系统状态 (state of the system) 非常有用。特别是，它允许您检查没有其他的包重定义了同一命令。

4.11 移动参数

已重新实现 (reimplemented) 处理 (即移动) 移动参数 (moving arguments) 时的保护设置 (setting of protect)，将信息从 `.aux` 文件写入其他文件 (如 `.toc` 文件) 的方法也已重新实现。详细信息可在文件 `ltdefns.dtx` 中找到。

新的描述
1994/12/01

我们希望这些更改不会影响其它包。

5 杂项命令，等

5.1 布局参数

`\paperheight`
`\paperwidth`

这两个参数通常由类设置为所用纸张的大小。这应该是实际的纸张大小，与 `\textwidth` 和 `\textheight` 不同，`\textwidth` 和 `\textheight` 是页边距内 (within the margins) 主要正文文本 (main text body) 的大小。

5.2 更改大小写

`\MakeUppercase {`⟨*text*⟩`}`
`\MakeLowercase {`⟨*text*⟩`}`

新的特性
1995/06/01

TEX 提供了两个原语 (primitives) `\uppercase` 和 `\lowercase` 来更改文本的大小写。它们有时用于文档类，例如，用所有大写字母来设置行文标题中的信息。

不幸的是，这些 TEX 原语不会改变 `\ae` 或 `\aa` 等命令访问的字符的大小写。为了克服这个问题，LATEX 提供了两个新命令 `\MakeUppercase` 和 `\MakeLowercase` 来完成这个任务。

举例：

`\uppercase{aBcD\ae\AA\ss\OE}`	ABCDæÅßŒ
`\lowercase{aBcD\ae\AA\ss\OE}`	abcdæÅßŒ
`\MakeUppercase{aBcD\ae\AA\ss\OE}`	ABCDÆÅ ŒE
`\MakeLowercase{aBcD\ae\AA\ss\OE}`	abcdæåßœ

`\MakeUppercase` 和 `\MakeLowercase` 命令本身很健壮，但它们有移动参数 (moving arguments)。

这些命令使用 TEX 原语的 `\uppercase` 和 `\lowercase`，因此具有许多意想不到的“特性 (features)”。特别是，它们更改了文本参数 (text argument) 中所有内容 (控件序列 [control-sequences] 名称中的字符除外) 的大小写：这包括数学 (mathematics)、环境名 (environment names) 和标签名 (label names)。

举例：

```
\MakeUppercase{$x+y$ in \ref{foo}}
```

生成 $X+Y$ 和警告：

```
LaTeX Warning: Reference `FOO' on page ... undefined on ...
```

从长远来看，我们希望使用全大写字体 (all-caps fonts)，而不是像 `\MakeUppercase` 这样的命令，但目前这是不可能的，因为这样的字体不存在。

为了使大/小写 (upper/lower-casing) 能够很好地工作，并且为了提供任何正确的连字符 (correct hyphenation)，LaTeX 2$_\varepsilon$ 必须在整个文档中使用相同的固定表 (fixed table) 来更改大小写。所使用的表是为字体编码 `T1` 而设计的；这与所有拉丁字母 (Latin alphabets) 的标准 TeX 字体一起工作得很好，但是在使用其他字母时会产生问题。

新的描述 1995/12/01

5.3 “book”类中的“openany”选项

`openany` 选项允许章 (chapter) 和类似的开头 (similar openings) 出现在左侧页面上。以前这个选项只影响 `\chapter` 和 `\backmatter`。它现在也影响 `\part`、`\frontmatter` 和 `\mainmatter`。

新的描述 1996/06/01

5.4 更佳的用户定义的数学显示环境

`\ignorespacesafterend`

假设您希望定义一个环境用于显示按公式编号的文本 (text that is numbered as an equation)，一种简单的方法如下：

新的特性 1996/12/01
新的描述 2003/12/01

```
\newenvironment{texteqn}
  {\begin{equation}
     \begin{minipage}{0.9\linewidth}}
    {\end{minipage}
  \end{equation}}
```

然而，如果您尝试过这种方法，那么您可能会注意到，在段落中间使用这种方法时效果并不理想，因为单词间的空格 (inter-word space) 出现在环境后的第一行的开头。

现在可以使用一个额外的命令 (名称很长) 来避免这个问题，应该像这里所示的那样插入该命令：

```
\newenvironment{texteqn}
  {\begin{equation}
     \begin{minipage}{0.9\linewidth}}
    {\end{minipage}
  \end{equation}
  \ignorespacesafterend}
```

此命令还有其他用途。

5.5 使间距正常

`\normalsfcodes`

这个命令用于恢复 (restore) 影响单词、句子等之间间距 (spacing) 的参数的正常设置 (normal settings)。

新的特性 1997/06/01

此功能的一个重要用途是纠正一个问题，据 Donald Arseneau 报道，当空格编码 (space codes) 的局部设置 (local setting) 生效时，页面标题中 (page headers) 的标点符号 (在所有已知的 TEX 格式中) 总可能是错误的。这些空格编码 (space codes) 可以通过例如 `\frenchspacing`) 命令和 verbatim 环境进行更改。

通常在 `\begin{document}` 中自动给出正确的定义，因此不需要显式设置；但是，如果在类文件中显式设置非空 (nonempty)，则自动默认设置 (automatic default setting) 将被覆盖。

6 升级 LaTeX 2.09 类和包

本节描述将现有的 LaTeX 样式 (style) 升级为包或类时可能需要进行的更改，但我们将从乐观模式 (optimistic mode) 开始。

许多现有的样式文件 (style files) 将使用 LaTeX 2$_\varepsilon$ 运行，而不需要对文件本身进行任何修改。当一切运行正常时，请在新创建的包或类文件中添加一个注释 (note)，以记录它使用新的标准 LaTeX 运行，然后将其分发给用户。

6.1 先试试吧！

你应该做的第一件事是在“兼容模式 (compatibility mode)”下测试你的样式 (style)。要做到这一点，您需要做的唯一更改就是将文件的扩展名更改为 `.cls`：只有当您的文件被用作主文档样式 (main document style) 时，才应该进行此更改。现在，在不做任何其他修改的情况下，对使用您文件的文档运行 LaTeX 2$_\varepsilon$。这假设您有一个合适的文件集 (collection of files)，用于测试您的样式文件提供的所有功能。(如果你还没有，现在是时候做一个了!)

Y 现在需要更改测试文档文件，使它们成为 LaTeX 2$_\varepsilon$ 文档：有关如何执行此操作的详细信息，请参阅 *LaTeX 2$_\varepsilon$ for Authors*【《LaTeX 2$_\varepsilon$ 作者指南》】，然后再试一次。您现在已经尝试了使用 LaTeX 2$_\varepsilon$ 本机模式 (native mode) 和 LaTeX 2.09 兼容模式 (compatibility mode) 测试文档。

6.2 排除故障

如果您的文件不能与 LaTeX 2$_\varepsilon$ 一起使用，可能有两个原因：

- LaTeX 现在有一个健壮的、定义良好的用于选择字体的设计者接口 (designer's interface)，这与 LaTeX 2.09 的内部版本有很大不同。
- 您的样式文件 (style file) 可能使用了一些已经更改或已经删除的 LaTeX 2.09 内部命令 (internal commands)。

在调试文件时，您可能需要比 LaTeX 2$_\varepsilon$ 通常显示的信息更多的信息。这是通过将计数器 `errorcontextlines` 的默认值从 -1 重置为更高的值例如 999 来实现的。

6.3 适应兼容模式

有时候，现有的 LaTeX 2.09 文档集 (collection) 使得完全放弃旧命令变得不方便或不可能。如果是这种情况，那么就有可能通过对以兼容模式 (compatibility mode) 处理的文件作出特别规定来兼容这两种约定 (conventions)。

`\if@compatibility`

当文档以 `\documentstyle` 而不是 `\documentclass` 类开头时，将设置此开关 (switch)。可以为任何一种情况提供适当的代码，如下所示：

```
\if@compatibility
  ⟨code emulating LaTeX 2.09 behavior⟩
\else
  ⟨code suitable for LaTeX2e⟩
\fi
```

6.4 字体命令

一些字体 (font) 和尺寸 (size) 命令现在由文档类 (document class) 而不是 LaTeX 内核定义。如果要将 LaTeX 2.09 文档样式 (document style) 升级为不加载标准类 (standard classes) 之一的类，则可能需要为这些命令添加定义。

`\rm \sf \tt \bf \it \sl \sc`

这些短格式字体选择命令 (short-form font selection commands) 都没有在 LaTeX 2_ε 内核中定义。它们由所有标准类文件 (standard class files) 定义。

如果您想在类文件中定义它们，有几种合理的方法可以做到这一点。

一个可能的定义是：

```
\newcommand{\rm}{\rmfamily}
...
\newcommand{\sc}{\scshape}
```

这将使字体命令正交 (orthogonal)，例如 `{\bf\it text}` 将生成粗体斜体 (bold italic)，即 ***text***。如果在数学模式 (math mode) 中使用，也会使它们产生错误。

另一个可能的定义是：

```
\DeclareOldFontCommand{\rm}{\rmfamily}{\mathrm}
...
\DeclareOldFontCommand{\sc}{\scshape}{\mathsc}
```

这将使 `\rm` 在文本模式 (text mode)(请参见上文) 下的行为类似于 `\rmfamily`，并将使 `\rm` 在数学模式 (math mode) 下选择 `\mathrm` 数学字母表 (math alphabet)。

因此 `${\rm math} = X + 1$` 将生成“$\mathrm{math} = X + 1$”。

如果不希望字体选择是正交的 (orthogonal)，则可以按照标准的类定义：

```
\DeclareOldFontCommand{\rm}{\normalfont\rmfamily}{\mathrm}
...
\DeclareOldFontCommand{\sc}{\normalfont\scshape}{\mathsc}
```

这意味着，例如 `{\bf\it text}` 将生成中等粗 (medium weight)(而不是粗) 斜体，即 *text*。

`\normalsize`
`\@normalsize`

保留了 `\@normalsize` 命令，以便与可能使用其值的 LaTeX 2.09 宏包兼容；但在类文件 (class file) 中重新定义它将没有任何效果，因为它总是被重置为与 `\normalsize` 具有相同的含义。

这意味着类现在必须重新定义 `\normalsize`,而不是重新定义 `\@normalsize`；例如 (一个相当不完整的定义)：

```
\renewcommand{\normalsize}{\fontsize{10}{12}\selectfont}
```

请注意，`\normalsize` 由 LaTeX 内核定义为错误消息。

`\tiny \footnotesize \small \large`
`\Large \LARGE \huge \Huge`

内核中没有定义其他“标准 (standard)”尺寸更改命令 (size-changing commands)：如果需要，每个命令都需要在类文件 (class file) 中定义。它们都由标准类 (standard classes) 定义。

\newcommand for the other size-changing commands. 这意味着您应该对 \normalsize 使用 \renewcommand，对其他尺寸更改命令 (size-changing commands) 使用 \newcommand。

6.5 过时的命令

有些宏包不能使用 LaTeX 2ε，通常是因为它们依赖于一个内部的 LaTeX 命令，而这个命令从来没有被支持过，现在已经被更改或者删除了。

在许多情况下，现在将有一种健壮的 (robust)、高级的 (high-level) 方法来实现以前需要的低级命令 (low-level commands)。请参考第 4 节，看看您现在是否可以使用 LaTeX 2ε 类和包编写者的命令。

当然，如果您的包或类重新定义了任何内核命令 (即在文件 latex.tex、slitex.tex、lfonts.tex、sfonts.tex 中定义的命令)，那么您需要根据新内核非常仔细地检查它，以防实现 (implementation) 发生更改或该命令不再存在于 LaTeX2e 内核中。

太多的 LaTeX 2.09 内部命令 (internal commands) 被重新实现或删除，无法在此列出所有命令。您必须检查已使用或更改的内容。

然而，我们将列出一些不再受支持的更重要的命令：

```
\tenrm \elvrm \twlrm ...
\tenbf \elvbf \twlbf ...
\tensf \elvsf \twlsf ...
   ⋮
```

此表单 (form) 的 (大约)70 个内部命令不再定义。如果您的类或包使用它们，请用 *LaTeX 2ε Font Selection*【《LaTeX 2ε 的字体选择》】中描述的新字体命令 (new font commands) 替换它们。

例如，应将 \twlsf 命令替换为：

```
\fontsize{12}{14}\normalfont\sffamily\selectfont
```

另一种可能性是使用 rawfonts 宏包，如在 *LaTeX 2ε for Authors*【《LaTeX 2ε 作者指南》】中所述的那样。

此外，请记住，许多由 LaTeX 2.09 预加载的 (preloaded) 字体不再预加载。

`\vpt \vipt \viipt …`

这些是 LaTeX 2.09 中的内部尺寸选择命令 (internal size-selecting commands)。(它们仍可以在 LaTeX 2.09 兼容模式下使用。) 请改用 `\fontsize` 命令，更详细的信息请参阅 *LaTeX 2ε Font Selection*【《LaTeX 2ε 的字体选择》】。

例如，应将 `\vpt` 替换为：

```
\fontsize{5}{6}\normalfont\selectfont
```

`\prm, \pbf, \ppounds, \pLaTeX …`

LaTeX 2.09 使用了几个以 `\p` 开头的命令，以提供“保护 (protected)”命令。例如，`\LaTeX` 被定义为 `\protect\pLaTeX`，而 `\pLaTeX` 被定义为生成 LaTeX 徽标。这使得 `\LaTeX` 变得健壮，尽管 `\pLaTeX` 并不健壮。

这些命令现在已使用 `\DeclareRobustCommand`(如第 4.10 节所述) 重新实现。如果您的包重新定义了其中一个 `\p`-命令，则必须删除重新定义，并使用 `\DeclareRobustCommand` 重新定义非-`\p` 命令。

`\footheight`
`\@maxsep`
`\@dblmaxsep`

LaTeX 2ε 不使用这些参数，因此已将其删除，LaTeX 2.09 兼容模式除外。类不再设置它们。

参考文献

[1] Donald E. Knuth. *The TeXbook*. Addison-Wesley, Reading, Massachusetts, 1986. Revised to cover TeX3, 1991.

[2] Leslie Lamport. *LaTeX: A Document Preparation System*. Addison-Wesley, Reading, Massachusetts, second edition, 1994.

[3] Frank Mittelbach and Michel Goossens. *The LaTeX Companion second edition*. With Johannes Braams, David Carlisle, and Chris Rowley. Addison-Wesley, Reading, Massachusetts, 2004.

LaTeX 2ε 汇总表：更新旧样式

以下节参考了 *LaTeX 2ε for Class and Package Writers*【《用于类和包作者的 *LaTeX 2ε*》】：

1. 它是一个类还是一个包？请参阅第 2.3 节了解如何回答这个问题。
2. 如果它使用另一个样式文件 (style file)，则需要获取该另一个文件的更新版本。有关如何加载其他类和包文件的信息，请参阅第 2.7.1 节。
3. 先试试吧！请参阅第 6.1 节。
4. 它有效吗？很好，但为了使您的文件成为结构良好的 LaTeX 2ε 文件，您可能还需要更改一些内容，使该文件既健壮 (robust) 又可移植 (portable)。所以你现在应该阅读第 2 节，特别是第 2.7 节。你也会在第 3 节中找到一些有用的例子。

   如果您的文件设置了新的字体、字体更改命令或符号，您还应该阅读 *LaTeX 2ε Font Selection*【《LaTeX 2ε 的字体选择》】。
5. 它不起作用吗？这里有三种可能性：

   - 错误信息是在读取文件时产生的；
   - 错误信息是在处理测试文件时产生的；
   - 没有错误，但是输出不是它应该的样子。

   别忘了仔细检查最后一种可能性。

   如果您已经到了这个阶段，那么您需要阅读第 6 节，以找到使您的文件工作的解决方案。