# 写给 LaTeX 2ε 类与宏包的作者


翻译: laughcry laughcry2002@163.com


12 March 1999

## 目录

# 1 引言

本文主要介绍与创作 LaTeX 中的类 (class) 与宏包 (packages) 相关的事项，特别强调了在将宏包从 LaTeX2.09 升级至 LaTeX $2_\varepsilon$ 所要注意的事项。关于后者，读者也可以参阅 Johannes Braams 发表在 TUGboat 15.3 上的文章内容。

## 1.1 为 LaTeX 2ε 编写类与宏包

LaTeX 是一个文档准备系统，它可以使作者将主要精力集中于文档的内容上，而无需对格式考虑过多。举例来说，书籍的章标题只需写成 `\chapter{`⟨*title*⟩`}` 而不用去指定诸如“三号黑体”之类的格式信息。

文档类 *(class)* 文件包含了如何将文档的逻辑结构（如 '`\chapter`'）进行格式化排版（如“三号黑体居中”）的相关信息。而另有一些功能与特征（例如，将文字彩色显示以及在文档中包含图形等功能）则包含在所谓的“宏包 *(package)*”文件中。

LaTeX2.09 与 LaTeX 2ε 之间最大的差别之一就在于用来写类与宏包的命令不完全相同。在 LaTeX2.09 中，由于缺少对编写 `.sty` 文件的专用命令的支持，许多作者不得不借助于更底层的 TeX 命令来完成特定的功能，因而极为不便。

LaTeX 2ε 则为宏包的写作提供了许多专用的高级命令。同时，在已有的类或宏包基础上构建新的类或宏包也变得更为简单，例如，可以基于 `article` 文档类为化工系编写一个新的专用技术报告文档类 `cetechr`。

## 1.2 概述

本文主要概要介绍如何编写 LaTeX 的类与宏包，我们没有介绍编写类与宏包所需的所有命令，有需求的读者可以参阅 *LaTeX: A Document Preparation System* 或 *The LaTeX Companion*。这里，我们只介绍类与宏包中引入的一些新命令的用法。

**第 2.7 节** 主要总结了写作类与宏包时的一些常用经验技巧。首先，我们对类与宏包的概念作了区分，给出了命令的命名约定，并介绍了 `doc` 和 `docstrip` 的用法, 同时解释了 TeX 的原语文件和 box 命令如何集成到 LaTeX 中去。此外，本节也对 LaTeX 文件的一般风格作了总结。

**第 3 节** 主要描述了类与宏包文件的结构。内容包含：如何在已有的类或宏包上构建新的类或宏包，如何声明可选项 (options) 及声明命令 (commands)。同时本节也包含有几个很好的例子。

**第 4 节** 列举了一些类与宏包中乃至的新命令。

**第 6 节** 对如何将已有的 LaTeX2.09 的类与宏包升级到 LaTeX 2ε 作了详细的介绍。

## 1.3 更多信息

关于对 LaTeX 的一般性介绍，包括 LaTeX 2$_\varepsilon$ 的新特性介绍，请阅读 Leslie Lamport 所著的 *LaTeX: A Document Preparation System* 一书 [3]。

关于 LaTeX 新特性的详细描述，以及包含有超过 150 个宏包的简要说明，可以参阅 Michel Goossens, Frank Mittelbach 和 Alexander Samarin 的 *The LaTeX Companion* 一书 [1]。

LaTeX 系统的基础是 TeX，关于 TeX 在 Donald E. Knuth 的经典著作 *The TeXbook* 中有最详细的描述 [2]。

此外，伴随着 LaTeX 的每份拷贝，也有许多相关的文档，每六个月发布一次的 *LaTeX News* 可以在 `ltnews*.tex` 文件中找到。作者指南 *LaTeX 2$_\varepsilon$ for Authors* 则主要描述了 LaTeX 的新特性，它包含在 `usrguide.tex` 文件中。在文件 `fntguide.tex` 中也是一份指南性文档 *LaTeX 2$_\varepsilon$ Font Selection*，它主要为类或宏包的作者提供了 LaTeX 的字体选择框架 (font selection scheme) 相关的信息。文件 `cfgguide.tex` 的指南 *Configuration options for LaTeX 2$_\varepsilon$* 主要讲述如何配置 LaTeX 等方面的内容，而文件 `modguide.tex` 中的指南 *Modifying LaTeX* 则主要陈述修订 LaTeX 背后的原则。

我们将逐步把 LaTeX 的源代码转换为一个 LaTeX 的文档：*LaTeX: the program*。这个文档包含了所有 LaTeX 命令的索引并可从文件`source2e.tex` 出发进行排版。

若想获取更多关于 TeX 及 LaTeX 的信息，敬请联系当地的 TeX 用户小组 (Users Group) 或国际 TeX 用户小组。这里提供几个有用的地址：

TeX Users Group, P.O. Box 1239, Three Rivers, CA 93271-1239, USA
Fax: +1 209 561 4584 Email: `tug@mail.tug.org`

UK TUG, 1 Eymore Close, Selly Oak, Birmingham B29 4LB, UK
Fax: +44 121 476 2159 Email: `uktug-enquiries@tex.ac.uk`

## 1.4 标准文档类的策略

在我们收到过的关于标准文档类的问题反馈中，大部分不是报告 bug 而是一些建议，这些建议的主旨通常是说，标准文档类中有一些功能设计得不尽合理，因此要求对标准文档类进行修改。

既然如此，为什么我们没有对标准文档类作修改呢？主要有如下几方面的原因：

- 尽管可能会引起误导，但大体上说，标准文档类的当前行为是符合其设计初衷的。
- 改变标准文档类并不是一种高明的做法，因为有无数的人在准备文档时要依赖于这些文档类。

因此，我们不会考虑修改标准文档类，甚至也不去浪费时间争论是否值得修改。当然，这并不意味着目前的标准文档类就完全没有缺点了，只是说目前有许多比“喋喋不休地解释标准文档类为什么不能修改”更值得去做的事罢了。

自然，我们也特别欢迎有更多、更好的新的文档类或宏包出现，以增强标准文档类的功能。因此，当你遇到标准文档类的缺点时，我们希望，你的第一想法是“我应该如何改进它的功能”。

关于对内核的改进需求也有类似的情形，许多功能不是直接在核中实现，而要留给各种文档类去实现。对内核的改动不论对你还是对于我们，都是一项很大的工程，因此较大的改进需要待在 LaTeX3 中实现。

# 2 编写类与宏包

本节简要介绍了编写 LaTeX 类与宏包相关的考虑。

## 2.1 老版本问题

如果你正要希望对存在的 LaTeX2.09 样式 (style) 文件进行升级，那么我们劝你最好冻结 2.09 版并放弃对它的继续维护。再像 1990 年代时建议的那样“为不同种版本的 LaTeX 同时进行维护”在目前已经是不可能的了。对于某些组织而言，对几种 LaTeX 版本的并行维护可能是必要的，但对于个人而言，应当摒弃过时的 LaTeX 版本，而仅使用最新版本的 LaTeX $2_\varepsilon$。

## 2.2 使用 ‘docstrip’ 和 ‘doc’ 程序

如果你正计划为 LaTeX 编写大型的类或宏包，那么你应当考虑使用 LaTeX 附带的 `doc` 软件。用这个程序编写的 LaTeX 类与宏包可以用两种方式来处理：

(1) 通过 LaTeX 处理得到文档；(2) 通过 docstrip 的处理得到 .cls 或 .sty 文件.

doc 软件可以自动生成各种定义的索引、命令的索引以及变更的抄本列表 (change-log lists)。它对于大型 TeX 源文件的维护及文档编制极为有用。

LaTeX 内核、标准文档类等的源文件都是 doc 文档，在发布中它们一般都位于 .dtx 文件中。实际上，你可以通过对 source2e.tex 运行 LaTeX，从而将 LaTeX 的内核排版成为一个长长的文档，并同时加上各种索引项。

关于 doc 与 docstrip 的更多信息，请参阅文件 docstrip.dtx, doc.dtx, 以及 *The LaTeX Companion*。关于其用法的例子，请参看 .dtx 文件。

## 2.3 究竟是用类还是用宏包?

在你希望给已有文档加一些新的 LaTeX 命令时，首先面临着一个问题：究竟是将这些新命令放在文档类 *(document class)* 中呢，还是放在宏包 *(package)* 中呢？通常选择的依据是：

> 如果这些命令可以用于各种文档类中，则将它们置于一个宏包中；
> 否则的话，宜于放它们置于文档类中。

文档类有两种：一种是类似 article, report 及 letter 一样独立的文档类；另一类则是在独立文档类的基础上扩展而来，例如 proc 文档类就是在 article 类的基础上扩展而来的。

举个例子来说，某公司在已有的 letter 文档类基础上建立了一个名为 ownlet 的文档类，用于在它们的稿纸的页眉上打印公司名称，由于新文档类不能与任何其它的文档类共用，因此我们将它做成一个 ownlet.cls 而不是 ownlet.sty。

相反的一个例子是 graphics 宏包，该宏包提供了一系列在文档中包含图形的命令。由于这些命令将用于各种文档类，因此使用了 graphics.sty 而不是 graphics.cls。

## 2.4 命令的命名

LaTeX 有三种类型的命令。

第一种是供普通作者用的命令 (author commands)，例如 `\section`, `\emph` 以及 `\times` 等。这类命令大多数都具有较短的名字，并且只使用小写形式。

第二种是供类与宏包的作者用的命令 (class and package writer commands)，这类命令的名字通常较长，并且具有大小写混合的形式，例如：

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

第三种是用于实现 LaTeX 本身的内部命令，如 `\@tempcnta`, `\@ifnextchar` 和 `\@eha` 等。这类命令的名字中大多包含 `@` 字符，它表示该命令只能用于类与宏包文件中，而不能用于通常的文档中。

然而，由于历史的原因，这些命令之间的区分经常变得模糊不清。例如，`\hbox` 属于内部命令，仅用于 LaTeX 内核；而命令 `\m@ne` 表示常数 $-1$ ，等同于 `\MinusOne`。

尽管如此，对上述三种命令作出区分还是非常有用的：如果命令的名字中含有 `@` 字符，则不应当把它是 LaTeX 语言所支持的命令，因为在将来的发布版本中，该命令有可能被更改。如果命令的名字是大小写混合形式，或在 *LaTeX: A Document Preparation System* 一书中有过专门的描述，则可根据 LaTeX $2_\varepsilon$ 的未来版本的情况决定是否支持这些命令。

## 2.5 盒子命令与彩色的使用

即使你不打算在自己的文档中使用彩色，也应仔细阅读本节内容，这些内容可以保证你的文档类或宏包与 `color` 宏包相兼容。这样那些使用彩色打印机的用户就可以很方便地使用你的这些类或宏包。

为能“安全地”使用彩色，最简单的方法是掌握如下原则：总是坚持使用 LaTeX 的盒子命令而不用 TeX 原语，使用 `\sbox` 而非 `\setbox`，使用 `\mbox` 而非 `\hbox`，使用 `\parbox` 或 `minipage` 环境而不使用 `\vbox`。由于 LaTeX 的盒子命令带有更多的可选项，这使得它们具有了与 TeX 一样强大的功能。

举个具体的例子，在命令序列 `{\ttfamily` ⟨*text*⟩`}` 中，字体会在 `}` 之前得到恢复，但在类似的结构 `{\color{green}` ⟨*text*⟩`}` 中，颜色则是在最后的 `}` 之后得到恢复的。通常，这种细微的差别不会影响排版效果，但若与 TeX 原语相结合使用时可能会有问题，如：

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

由于颜色的恢复是在 } 之后发生，所以原始颜色没有存储到盒子中去。至于这样的命令序列会出现什么问题，则跟具体的上下文有关：小到有可能导致后续的内容得到错误的颜色设置，大到直接引起打印时 dvi- 驱动出错等等。

另一个类似的例子是 \normalcolor 命令。通常情况下，它的实现只是一个 \relax 命令 (即，什么也不做) 但你可以像使用 \normalfont 那样将主文档的颜色值用于 captions 或 section headings 等页面部分区域的内容。

## 2.6 定义文本与数学字符 (Defining text and math characters)

由于 LaTeX 2ε 支持多种编码方案 (encodings)，因此必需为符号 (symbol)、重音 (accent)、composite glyphs 等的产生定义合适的命令（通过使用 *LaTeX 2ε Font Selection* 中介绍的命令），由于这部分内容仍然处于未成熟的发展阶段，因此在使用这些时应当特别谨慎。

此外，使用 \DeclareRobustCommand 命令也应当注意要做到与编码方案独立。

注意，在数据模式之处使用数学字体将不再支持，例如， \textfont 1 与 \scriptfont 2 这样的命令都不应该在其它模式中使用。

## 2.7 一般性的风格 (General style)

新系统提供了许多命令用于设计结构化良好、健壮性与可移植性俱佳的文档类和宏包。本节简要介绍如何用好这些命令。

### 2.7.1 装载其它文件 (Loading other files)

LaTeX 提供了如下命令 新描述 1995/12/01

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

用于在一个类/宏包中使用另一个类/宏包。我们强烈推荐使用这些命令，而不要使用原语级的 \input 命令。主要原因有以下几种。

用 \input ⟨*filename*⟩ 装载的文件将不会被列在 \listfiles 清单中。

如果总是用 \RequirePackage... 或 \usepackage 来装载类/宏包，不会出现重复装载，而用 \input 则不然，既浪费了时间与内存空间，又可能导致各种其它奇怪的结果。

如果一个宏包提供有选项，那么用 \input 装载时也可能出现奇异的结果，但使用\usepackage 或\RequirePackage... 则不会出现这种情况。

如果在宏包 foo.sty 中使用 \input baz.sty 来装载宏包 baz.sty，则用户会得到一个如下的告警：

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

综上所述，用 \input 来装载宏包绝对不是一个明智的做法。

不幸的是，如果你正在考虑将样式文件 myclass.sty 升级成为一个文档类文件，则你仍然必需考虑如何使那些用了 \input myclass.sty 命令的旧文档也能正常工作。

同样的情形也发生在标准文档类 (article, book 和 report) 上，那是因为大量已有的 LaTeX2.09 文件都使用了诸如 \input article.sty 这样的命令。目前标准文档类这种问题的解决办法是系统提供了一个非常简单的文件 article.sty, book.sty 以及 report.sty，其内容仅仅是装载相应的文档类文件。

举例来说， article.sty 文件的内容如下：

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

你可以效仿这种做法。或者，在足够安全时，甚至可以索性删除 myclass.sty 文件。

### 2.7.2 健壮性 (Make it robust)

作为一个良好的实践性做法，我们建议在编写类/宏包时总是使用 LaTeX 的命令。

基于这种建议，应该避免使用 \def... 命令，而要多使用 \newcommand, \renewcommand 或 \providecommand 等命令，同时 \CheckCommand 有时也很有用。这样做可以避免你在不经意间错误地重复定义了某个命令，从而导致不期望的错误结果。

如果要定义一个新的环境，建议使用 \newenvironment 或 \renewenvironment 而不要使用 \def\foo{...} 或 \def\endfoo{...}。

如果要设置或更改 ⟨*dimen*⟩ 或 ⟨*skip*⟩ 的取值，建议使用 \setlength。

为了对盒子进行操作，使用 LaTeX 命令，如 \sbox、\mbox 以及 \parbox，避免使用 \setbox、\hbox 或 \vbox。

使用 \PackageError、\PackageWarning、\PackageInfo (或其它等价的文档类命令)，而不要使用 \@latexerr、\@warning 或 \wlog 等命令。

虽然仍旧可以通过定义 \ds@⟨*option*⟩ 及调用 \@options 的方式来声明选项，但我们建议使用 \DeclareOption 和 \ProcessOptions 命令来完成同样的任务，它们具有更大的灵活性并具有更小的内存需求。因此，不要使用：

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

而应当使用：

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

以上这些建议的好处除了使你的代码具有更好的可读性外，更重要的是，可以保证你的类/宏包在 LaTeX 将来的版本中可以继续使用。

### 2.7.3 可移植性 (Make it portable)

保持文档类/宏包的可移植性同样是一个重要的问题，为了保证这一点，建议在文档中只使用可见的 7-bit 字符，文件名至多包含 8 个字符 (扩展名至多 3 个字符)。此外，特别注意不要与标准 LaTeX 发行版中的文件名相重复。

有时给自己的类/宏包提供一个公共的前缀也很有用，例如，University of Nowhere 的类/宏包使用前缀名 unw，这样可以避免每个大学都建立相同的文档类名字 thesis.cls 而出现重复。

如果你的文档类/宏包依赖于 LaTeX 内核或某个宏包的一些特性，则应当指定所需要的发行日期。例如，宏包错误信息处理的命令是在 June 1994 的发行中才引入的，如果你使用了这些错误信息处理命令，则应当加上如下的命令：

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

#### 2.7.4 有用的 hook(Useful hooks)

有些类/宏包必需对 `\document` 或 `\enddocument` 进行修改才能达到所需的效果，现在已经无需这么做了，你可以使用 'hooks' `\AtBeginDocument` 和 `\AtEndDocument` (参阅第 4.6 节) 来达到目的。此外，使用这些 hooks 还可以保证类/宏包对未来 LaTeX 版本的兼容性，以及顺利地与其他宏包协同工作。

注意，`\AtBeginDocument` 中的代码都将是序言中内容的一部分，因此对能放入其中的命令有一些限制，特别地，排版类的命令一般都不能放入其中。

新描述 1996/12/01

# 3 类/宏包的结构 (The structure of a class or package)

LaTeX $2_\varepsilon$ 类与宏包比 LaTeX2.09 的样式文件具有更大的结构化特性。其主要结构框架如下：

**标识节** 宣告自身是一个类或宏包，并给出简要描述性信息。

**初步声明节** 这里对一些命令进行声明，同时装载一些其它文件。通常，这个小节仅声明在后续的选项处理中将要使用的命令。

**选项节** 声明并处理可选项。

**更多声明节** 这是文件的主体部分，主要进行：声明新变量、新命令和新字体，装载其它文件。

## 3.1 标识节 (Identification)

类/宏包文件的首要任务是标识自身。宏包文件的识别部分格式为：

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

例如：

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

文档类文件的标识部分格式为：

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

例如：

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

新描述 1998/06/19

其中 ⟨*date*⟩ 的格式是 'YYYY/MM/DD'，并且当使用了可选参数时此部分不可省略 (`\NeedsTeXFormat` 命令也遵循同样的约定)。必需完全遵守这样的语法约定，任何微小的偏差都会导致低层的 TEX 错误 —the commands expect a valid syntax to speed up the daily usage of the package or class and make no provision for the case that the developer made a mistake!

当用户在他们的 `\documentclass` 或 `\usepackage` 命令中书写了日期时，这里的日期部分就会被检查。例如，如果你写了这样的语句：

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

那么用户就会得到一个告警信息说，他们的 `article` 版本已经过时了。

文档类的描述信息会在使用该类时显示出来，而宏包的描述信息则放进了抄本文件。此外，这些描述信息都可以通过 `\listfiles` 命令显示出来。短语 `Standard LaTeX` 仅用于标准 LATEX 发布中的文件的标识，而不应用于自定义的文件。

## 3.2 使用类与宏包 (Using classes and packages)

LaTeX2.09 的样式文件与 LaTeX $2_\varepsilon$ 的类/宏包文件的第一个主要差别是，LaTeX $2_\varepsilon$ 支持模块性 *(modularity)*，即可以由许多较小的建筑块构建更大的文件。

在 LaTeX 的类/宏包中装载另一个宏包使用如下的语法：

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

例如：

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

此命令与普通文档作者用的命令 `\usepackage` 具有相同的语法。它允许在类/宏包内使用其它宏包的特性。例如，通过装载 `ifthen` 宏包，类/宏包的作者就可以使用诸如 'if…then…else…' 的命令了。

一个 LaTeX 类中装载另一个类文件使用如下语法：

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

例如：

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

此命令与普通文档作者用的命令 `\documentclass` 具有相同的语法。它允许一个类具有另一个类提供的语法及外观。例如，通过装载 `article` 类，类的作者就可以只对 `article` 中不喜欢的部分作修改，而无需构建一个完整的新类。

当你希望在当前的类中以完全相同的选项装载其它的类/宏包时，可以使用以下命令：

新特性 1995/12/01

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

例如：

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

## 3.3 声明选项 (Declaring options)

新描述
1998/12/01

LaTeX2.09 的样式文件与 LaTeX $2_\varepsilon$ 类/宏包的另一个重要差别在于对选项的处理方式上。类/宏包可以声明选项，同时选项可由作者指定。例如在文档类 `article` 中声明有 `twocolumn` 选项。注意选项的名字也应当遵循 'LaTeX 的命名约定'，自然，其中不应当包含任何控制字符序列。

声明选项的格式为：

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

例如，在宏包 `graphics` 中声明选项 `dvips` 的代码如下（作了简化）：

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

这表明，当用户在文档中书写了 `\usepackage[dvips]{graphics}` 时，将会装载 `dvips.def` 文件。另一个例子是在文档类 `article` 中声明了选项 `a4paper`，用于设置 `\paperheight` 与 `\paperwidth` 的长度值：

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

如果用户指定了一个类/宏包中未加声明的选项，那么在缺省情况下，会产生一个告警 (对于类而言) 或错误 (对于宏包而言)。正确的做法是：

```
\DeclareOption*{⟨code⟩}
```

例如，为了能使宏包 `fred` 对未知的选项产生一个告警而不是错误信息，可以这样书写：

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

这样，当用户使用了 `\usepackage[foo]{fred}` 这样的语句时，他们会得到一个告警信息 `Package fred Warning: Unknown option `foo'.` 再举一例，当使用选项 ⟨*ENC*⟩ 时希望宏包 `fontenc` 会装载文件 ⟨*ENC*⟩`enc.def`，这可以实现为：

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

另外，还可以通过命令 \PassOptionsToPackage 或 \PassOptionsToClass，将选项传递给另一个类/宏包（注意它们属于专门设计的操作，仅能用于处理选项名字）。例如，要将每一个未知的选项都传递给 article 文档类，可以使用： 新描述 1998/12/01

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

注意应当在这段代码的后续代码中确保装载相应的文档类，否则该选项将永远也得不到处理。

至此，我们已经解释了如何声明选项，但还没介绍如何执行它们。如果文档调用了这些选项，则相应的选项处理程序可以用如下格式加以调用：

```
\ProcessOptions\relax
```

这将会执行每个指定（且声明过了的）选项的 ⟨*code*⟩ 部分 (细节部分请参阅第 4.7 节)。

例如，若宏包 jane 包含了：

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

又若用户文档中使用了 \usepackage[foo,bar]{jane}，那么用户将看到消息文本 Saw foo. 以及 What's bar?

## 3.4 最简单的文档类示例 (A minimal class file)

文档类/宏包的主要工作是定义新命令或更改命令的行为，这些工作主要放在类/宏包的主体内，并使用诸如 \newcommand 或 \setlength 命令实现。

LaTeX 2ε 也提供了一些新的命令以辅助作者缩写类/宏包，这些命令的细节请参阅第 4 节。

有三项基本的内容是每一个文档类文件中必需包含的： \normalsize 的定义、\textwidth 的取值和 \textheight 的取值。因此，一个最简单的文档类文件[1]是这样的：

```
\NeedsTeXFormat{LaTeX2e}
\ProvidesClass{minimal}[1995/10/30 Standard LaTeX minimal class]
\renewcommand{\normalsize}{\fontsize{10pt}{12pt}\selectfont}
\setlength{\textwidth}{6.5in}
\setlength{\textheight}{8in}
```

这个文档类不支持脚注 (footnotes)、边注 (marginals)、浮动体 (floats) 等，也没有提供任何诸如 \rm 的两字母命令，因此，实际上大部分的文档类都要远比这个“迷你型”的文档类复杂得多。

## 3.5 范例：个性化的信笺类 (Example: a local letter class)

公司一般都有自己的个性化的信笺用文档类。本节实现了这样一个简单的实例。

该文档类首先宣告自身的标识为 neplet.cls。

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

然后接受任何传递给 letter 的选项，并同时使用 a4paper 作为默认选项装载文档类 letter。

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

为了使用公司的信头，对页面样式 firstpage 进行了重新定义。firstpage 通常用于设置信笺的首页的风格样式。

[1]此文档类目前已经成为标准发布的一部分，文件名为 minimal.cls。

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

至此，大功告成！

## 3.6 范例：新闻稿纸文档类 (Example: a newsletter class)

可以通过对文档类 `article` 作更改，可以用 LaTeX 来排版简单的新闻。同样，文档类首先识别自身为 `smplnews.cls`。

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

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

其次，文档类可以接受 `article` 的大部分选项，但不接受 `onecolumn` 选项，该选项被关闭掉；另外，可以接受 `green` 选项，用于将信头设置为绿色。

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

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

\ProcessOptions\relax
```

然后，用选项 `twocolumn` 装载文档类 `article`。

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

由于新闻稿纸要用彩色打印，因此，文档类装载了宏包 `color`。另外，没有在此文档类中指定设备驱动选项，用户可以在使用此文档类时指定之。

```
\RequirePackage{color}
```

接下来，文档类重定义了 `\maketitle` 以产生合适的格式 (72pt Helvetica bold oblique with colour)。

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

对 \section 重定义以关闭其自动编号功能。

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

\setcounter{secnumdepth}{0}
```

同样，它也设置了三项基本内容。

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

实践中，文档类实现的功能远远超过这些，如，提供期号、作者、页面布局等方面的命令。以上例子可以作为一个简单的范本，实际上，文档类 ltnews 的实现并不比这个例子复杂多少。

# 4 类/宏包中常用命令 (Commands for class and package writers)

本节简要介绍可在文档类/宏包中使用的新命令，更多细节请阅读 *LaTeX: A Document Preparation System*, *The LaTeX Companion* 以及 *LaTeX 2ε for Authors* 等。

## 4.1 标识 (Identification)

这里讨论的第一组命令用于标识类/宏包。

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

该命令告诉 TEX：此文件应当使用名为 ⟨*format-name*⟩ 的格式进行处理。可以使用可选参数 ⟨*release-date*⟩ 进一步指定所需格式的最早发行日期。当机器上的格式文件发行日期早于该指定时，就会产生一条告警信息。标准的 ⟨*format-name*⟩ 是 `LaTeX2e`。如果指定日期的话，其格式必需是 YYYY/MM/DD。

示例：

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

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

该命令声明，当前的文件包含了为文档类 ⟨*class-name*⟩ 或宏包 ⟨*package-name*⟩ 提供的定义命令。

如果使用了可选的 ⟨*release-info*⟩，则必需包括：

- 发行日期，格式为 YYYY/MM/DD；
- （可选）接续一个空白符及简短描述信息，通常包含一个版本号。

以上语法必需严格遵守，因为这些信息将会被 `\LoadClass`、`\documentclass` (对于文档类而言)，或 `\RequirePackage`、`\usepackage` (对于宏包) 使用，以便测试发行日期是否过时。

整个 ⟨*release-info*⟩ 信息会被 `\listfiles` 命令显示出来，因此内容不宜过长。

示例：

```
\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*⟩`]`

这条命令与前两条命令相类似，只是这里需要提供完整的文件名（包括扩展名）。该命令用于声明主文档类/宏包之外的任何文件。

示例：

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

注意，短语 `Standard LaTeX` **不应当**出现在自定义的文件的标识文字中。

## 4.2 装载文件 (Loading files)

这一组命令用于在已有的文档类/宏包基础上创建新的文档类/宏包。 **新特性** 1995/12/01

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

文档类/宏包应当用这些命令来装载其它的宏包。

命令 `\RequirePackage` 的用法与普通文档用的命令 `\usepackage` 的用法相同。

示例：

```
\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，而不能用于宏包文件，而且在一个类文件内最多只能使用一次。 **新特性** 1995/12/01

命令 `\LoadClass` 的用法与 `\documentclass` 的用法相同，都用来装载一个类文件。

示例：

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

以上两个带有 `WithOptions` 的命令在装载类（或宏包）时将仅使用当前文件（类或宏包）所使用的选项。详细用法，参见后文的第 4.5 节。

新特性 1995/12/01

## 4.3 选项声明 (Option declaration)

以下命令用于选项的声明及处理。注意，每个选项的名称必需遵循 'LaTeX 命名约定'。

新描述 1998/12/01

其中，有一些命令是专门为以下命令中的 ⟨*code*⟩ 参数而设计的（详见下文）。

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

该命令将 ⟨*option-name*⟩ 声明为所在文件（类或宏包）的一个选项。

⟨*code*⟩ 中包含了有效的 LaTeX $2_\varepsilon$ 命令序列。若当前文件（类或宏包）使用了 ⟨*option-name*⟩ 选项时，将会执行 ⟨*code*⟩ 中的命令序列。

示例：

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

`\DeclareOption* {`⟨*code*⟩`}`

该命令用来指定类或宏包使用未加声明的选项时要执行的命令序列 ⟨*code*⟩，因此命令序列 ⟨*code*⟩ 也常称为缺省选项代码。

如果在一个文档类中没有使用 `\DeclareOption*` 命令，那么所有未加声明的选项将会传递给所有宏包（就像那些声明过的选项一样）。

如果在一个宏包中没有使用 `\DeclareOption*` 命令，那么所有未加声明的选项将会产生一个错误信息。

## 4.4 选项代码中的专用命令 (Commands within option code)

有两条专门设计用于 `\DeclareOption` 和 `\DeclareOption*` 的 ⟨*code*⟩ 参数中的命令。

`\CurrentOption`

该命令用于将当前选项的名称展开。

`\OptionNotUsed`

该命令用于将当前选项加入到 '未使用的选项 (unused options)' 列表中去。

目前，你可以在这些 ⟨*code*⟩ 参数中直接使用 (#) 符号而无需任何特殊处理（在此之前，这个符号是需要连续双写的）。 新特性 1995/06/01

## 4.5 传递选项 (Moving options around)

以下两条命令也经常用于 `\DeclareOption` 与 `\DeclareOption*` 的 ⟨*code*⟩ 参数中：

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

命令 `\PassOptionsToPackage` 将选项列表 ⟨*options-list*⟩ 中的各个选项名传递给宏包 ⟨*package-name*⟩，即，将 ⟨*option-list*⟩ 加入到以后用 `\RequirePackage` 或 `\usepackage` 装载的宏包 ⟨*package-name*⟩ 的选项列表中。

示例：

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

也等同于：

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

类似地，`\PassOptionsToClass` 命令用在文档类中，作用是将选项传递给另一个文档类（用 `\LoadClass` 装载）。

在效果与用法上，这两条命令可与以下两条相比较 (参见第 4.2 节中的相关说明): 新描述 1995/12/01

```
\LoadClassWithOptions
\RequirePackageWithOptions
```

命令 \RequirePackageWithOptions 类似于 \RequirePackage，但前者在装载宏包时，使用与当前文件（类或宏包）完全相同的选项，而且不能显式提供或用 \PassOptionsToPackage 命令来传递选项。

命令 \LoadClassWithOptions 的主要目的是用来方便地基于一个文档类构造另一个文档类，例如：

```
\LoadClassWithOptions{article}
```

在效果上，它基本上等同于以下的构造方法

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

显然，第一种写法要简洁得多，而且使用 \LoadClassWithOptions 命令运行起来更快捷一点。

然而，如果文档类也声明了自己的专有选项，两种构造方法效果则不相同。试比较：

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

和

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

假设调用当前文档类的文档中使用了 landscape 选项，那么在前一个例子中，会同时以此选项选项装载类 article，而在后一个例子中，装载 article 时则不会附带 landscape 选项（因为选项 landscape 是经过声明的，而装载类 article 的行为只发生在缺省选项的处理代码中）。

## 4.6 延迟的代码 (Delaying code)

这里的前两条命令也主要是用于命令 `\DeclareOption` 或 `\DeclareOption*` 的 ⟨*code*⟩ 参数中。

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

该命令用于声明代码并内部保存代码 ⟨*code*⟩，并在处理完整个的类/宏包文件后，执行这些代码。

该命令允许重复出现，此时代码部分将会依其出现次序保留并被执行。

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

该命令将代码 ⟨*code*⟩ 保留下来，并在 LaTeX 执行到 `\begin{document}` 或 `\end{document}` 时执行这些代码。

`\AtBeginDocument` 中的代码部分 ⟨*code*⟩ 执行的时机为：`\begin{document}` 行将执行完毕、字体选择表 (font selection tables) 建立之后，此时所有与排版相关的工作已经准备妥当，当前字体即为 normal font。

`\AtBeginDocument` 中不应当书写与排版有关的命令，否则排版效果难以预测。

新描述 1995/12/01

`\AtEndDocument` 中的代码 ⟨*code*⟩ 执行的时机为：`\end{document}` 命令执行之初、文档最末页完成之前、以及剩余浮动环境处理之前。如果希望 ⟨*code*⟩ 中的一部分代码在最末页完成之后或剩余的浮动环境处理之后执行，那么应该在代码中的适当位置加上 `\clearpage`，以达到预期的效果。

该命令同样允许重复出现，代码部分将会依其出现次序保存并执行。

`\AtBeginDvi {`⟨*specials*⟩`}`

新特性 1994/12/01

该命令用于在一个盒子中保存内容，并在 `.dvi` 文件第一页的 'shipout' 首部写入这些内容。

不能在 `.dvi` 文件中写入任何与排版有关的内容。

该命令允许重复出现。

## 4.7 选项处理 (Option processing)

`\ProcessOptions`

该命令将会执行所使用的各选项的代码部分 ⟨*code*⟩。

首先，介绍 `\ProcessOptions` 在宏包文件中如何工作，然后再讨论在类文件中的情形。

要理解 `\ProcessOptions` 在宏包文件中的详细工作机制，必需能够正确区分*局部选项*和*全局选项*这两个概念。

- **局部选项 (Local options)** 是指显式地用以下命令的 ⟨*options*⟩ 参数指定的选项：

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

- **全局选项 (Global options)** 是指所有其它的、用户在文档中用 `\documentclass[`⟨*options*⟩`]` 指定的选项 ⟨*options*⟩。

例如，若某个文档开头的片段为：

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

同时宏包 `gerhardt` 调用了另一宏包 `fred`：

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

则：

- `fred` 的局部选项为 `german`、`dvips`、`a4paper` 和 `errorshow`；
- `fred` 的全局选项只有一个 `twocolumn`。

当调用了 `\ProcessOptions` 时，会依次进行如下处理：

- 首先，对于 fred.sty 中用 \DeclareOption 声明过了每个选项，如果该选项对于 fred 而言是全局选项或局部选项，则执行相应的代码段。

  对各选项处理的顺序与它们在 fred.sty 中声明的次序相一致。

- 其次，对于每个剩余的局部选项，如果在某个地方定义了 \ds@⟨*option*⟩ 命令（在 \DeclareOption 中定义的情形除外），则会执行之；否则，则执行 '缺省选项处理代码'。如果没有指定缺省选项处理代码，则产生一条错误消息。

  以上处理的次序也与各选项指定的次序相一致。

整个处理过程中，系统会保证每个选项对应的代码至多执行一次。

仍以前面的代码为例，如果 fred.sty 中包含有：

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

则文档处理的结果将会是：

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

请注意以下几点：

- 选项 dvips 的代码的执行要先于选项 german 的代码，这与 fred.sty 中的选项声明次序是相吻合的；
- 在处理过程中，选项 german 的代码只执行了一次；
- a4paper 与 errorshow 两个选项均按照 \DeclareOption* 中的处理（依指定的次序）产生了一条错误消息，但 twocolumn 选项并没有按照 \DeclareOption* 中的处理，这时因为 twocolumn 是一个全局选项而非局部选项。

在文档类文件中，\ProcessOptions 的工作机制也大体类似，只有两点不同：所有的选项均是局部选项，\DeclareOption* 的缺省值是 \OptionNotUsed，而不是产生一条错误消息。

注：由于 \ProcessOptions 命令还有一种星号形式的写法，因此一般要在非星号形式后紧接一个 \relax（就像前面的例子中那样），这样可以阻止不必要的命令超前读取，避免可能由此带来的误导性出错消息。

新描述 1995/12/01

\ProcessOptions* \@options

该命令与 \ProcessOptions 相类似，但在选项的处理次序上，它与调用命令中选项的指定次序相一致，而不是依照类/宏包中选项的声明次序。对宏包而言，这将意味着，全局选项会优先得到处理。

为了减轻将旧的样式文档升级为 LaTeX 2$_\varepsilon$ 类文件时的工作量，来源于 LaTeX2.09 的命令 \@options 也被进行了改造，其功能完全等同于 \ProcessOptions*。

\ExecuteOptions {⟨*options-list*⟩}

对于 ⟨*options-list*⟩ 中的每个 option，会依次序地执行命令 \ds@⟨*option*⟩ (如果未定义此命令，则忽略之)。

该命令置于 \ProcessOptions 之前，用于提供一个 '缺省选项列表'。例如：如果希望在类文件中使用“双面打印，11 磅的字体，分两栏”的布局，则可以用：

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

## 4.8 安全处理文件的命令 (Safe file commands)

这几条命令都是用于处理文件的输入的，使用它们可以保证，即使在所需文件不存在的情况下，也能得到用户友好的处理办法。

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

如果文件存在，则执行 ⟨*true*⟩ 中的代码，否则执行 ⟨*false*⟩ 中代码。

该命令本身并不输入文件的内容。

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

如果文件 ⟨*file-name*⟩ 存在，则执行 ⟨*true*⟩ 中的代码，然后将文件内容输入进来。

如果文件不存在，则执行 ⟨*false*⟩ 中的代码。

本条命令是借助 `\IfFileExists` 来实现的。

## 4.9 错误报告及其它 (Reporting errors, etc)

这些命令主要用于第三方的类/宏包，作用是报告错误或向作者提供信息等。

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

这两条命令用于产生一条错误消息。显示错误消息 ⟨*error-text*⟩ 及错误提示符 `?`，如果用户输入了 `h` 的话，还会进一步显示 ⟨*help-text*⟩。

在 ⟨*error-text*⟩ 和 ⟨*help-text*⟩ 中：`\protect` 用于终止命令的展开；`\MessageBreak` 用于产生换行；`\space` 则产生一个空白符。

注意 ⟨*error-text*⟩ 会自动停顿下来，因此不必再人为实现此效果。

示例：

```
\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.
}
```

显示的结果为：

```
! Package ethel Error: Your hovercraft is full of eels,
```

```
(ethel)                and \foo is FOO.

See the ethel package documentation for explanation.
```

若用户此时输入了 h，则显示：

```
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*⟩}

以上四个 Warning 型的命令与错误命令相类似，不同之处在于，它们产生的是告警消息，而非错误消息，因而不会出现错误提示符 (?)。

其中前两个 Warning 命令会同时显示产生告警的行号，而后两个 WarningNoLine 命令则不显示行号信息。

最后两个 Info 命令的行为也类似，但它只是将信息输出到抄本文件中。输入的信息总有带有行号的，这两条命令没有对应的 NoLine 版本。

在 ⟨*warning-text*⟩ 和 ⟨*info-text*⟩ 中：\protect 用于终止命令的展开；\MessageBreak 用于产生换行；\space 则产生一个空白符。同样，不需要为它们指定结束行为，因为在默认的实现中，已经包含了这样的行为。

## 4.10 定义命令 (Defining commands)

LaTeX 2ε 提供了几种在类/宏包中定义（或重定义）命令的方法。

以下带星号形式的命令可用于定义那些非 long 的命令 (TeX 的术语)，这将对命令的错误处理很有用，除非那些命令的参数是整个文本段的情况。

新特性 1994/12/01

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

此命令的参数与 \newcommand 完全相同，但它用来声明一个健壮的 (robust) 命令，即使当 ⟨*definition*⟩ 中含有脆弱的 (fragile) 代码时也如此。可以使

用这个命令定义新的健壮型命令，或重定义已存在的命令并使它成为健壮型的命令。若一个命令被重复定义的话，会在抄本中写入相关的信息。

例如，若 \seq 的定义为：

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

则命令 \seq 可用作移动参数 (moving arguments)，尽管单独这样用 \ifmmode 是不行的。例如：

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

同时注意，在上述定义的开头处的 \ifmmode 之前不必使用 \relax 命令 (\relax 主要是防止在不适当的时候对命令进行展开)，这是因为内部已经提供了对命令展开的保护机制。

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

此命令与 \newcommand 的参数完全相同，但它不定义命令 ⟨*cmd*⟩，而只检查命令 ⟨*cmd*⟩ 的当前定义是否与 ⟨*definition*⟩ 完全吻合，如果不相吻合，则会引发一条错误消息。

当你打算修改某个命令的定义时，用这个命令进行事先检查将很有用，它可以帮你检查是否有其它的宏包对该命令作了重定义。

## 4.11 移动参数 (Moving arguments)

处理移动参数时进行的保护设置已经被重新实现过，主要表现在信息写入的文件已经从 .aux 换为了其它文件（如 .toc 等）。细节请参阅文件 ltdefns.dtx。

新描述 1994/12/01

我们希望这种变化不会影响太多的宏包文件。

# 5 其它命令 (Miscellaneous commands, etc)

## 5.1 布局参数 (Layout parameters)

`\paperheight`
`\paperwidth`

提供这两个参数主要是为了在类中定义实际的纸张大小，而另两个参数 `\textwidth` 及 `\textheight` 则是指纸张边界之内的文字部分的大小。

## 5.2 大小写更改 (Case changing)

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

TEX 提供了两个原语级的命令 `\uppercase` 和 `\lowercase` 用于更改文本的大小写，它们也经常在文档类中使用，例如将 running head 中的文本设置为大写形式等。

新特性 1995/06/01

遗憾的是，这两条原语级的 TEX 命令并不会改变诸如 `\ae` 或 `\aa` 这样的字符的大小写。为此，LATEX 提供了两个新命令 `\MakeUppercase` 和 `\MakeLowercase` 用于解决这样的问题。

示例：

| | |
|---|---|
| `\uppercase{aBcD\ae\AA\ss\OE}` | ABCDæÅßŒ |
| `\lowercase{aBcD\ae\AA\ss\OE}` | abcdæÅßŒ |
| `\MakeUppercase{aBcD\ae\AA\ss\OE}` | ABCDÆÅSSŒ |
| `\MakeLowercase{aBcD\ae\AA\ss\OE}` | abcdæÅßœ |

`\MakeUppercase` 和 `\MakeLowercase` 这两条命令本身不是健壮的 (robust)，但它们都可以拥有移动参数。

同时这两条命令的实现中使用了 TEX 原语级命令 `\uppercase` 和 `\lowercase`，因此，会出现一些意想不到的特性。比如，这两条命令会将参数中的所有文本（命令控制序列的名称除外）变成大（小）写，包括数学部分 (mathematics)、环境名 (environment names) 以及标签名 (label names) 等等。

例如：

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

的结果是生成了 $X+Y$ 及告警消息：

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

长远看，我们应该使用全大写的字体，而不是像现在这样使用 \MakeUppercase 命令。但在目前只能如此，因为还不存在这样的字体。

为了保证大小写工作正常以及正确地进行断词, LaTeX 2$_\varepsilon$ 必需在整篇文档中借助一个固定的表来更改大小写。由于这个表采用了 T1 的字体编码，因此，它与标准的 TeX 字体在拉丁字符表中配合得很好，但在其它字符表中可能会有问题。

新描述 1995/12/01

## 5.3 'book' 文档类中的 'openany' 选项

选项 openany 的作用是允许将章 (chapter) 一级标题放在左首页面上，以前版本中此命令只影响 \chapter 和 \backmatter，现在它将还同时影响到 \part、\frontmatter 和 \mainmatter。

新描述 1996/06/01

## 5.4 更好的用户自定义数学环境

\ignorespacesafterend

假若你希望为一些文本定义一个像数学公式那样可以自动编号的环境的话，最直接的做法可能是：

新特性 1996/12/01

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

然而，你将会发现，它不能很好地工作，尤其是在一个段落中间使用时，会有一个字间的空白出现在该环境后面文本的第一行行首。

现在，提供了一个额外的命令 (名字很长哦) 可以解决这个问题，即使用如下的代码：

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

### 5.5 Normalising spacing

`\normalsfcodes`

此命令是为了将那些影响字间、段间空白的参数恢复为通常的取值而引入的。

新特性 1997/06/01

此命令的重要应用是为了纠正一个由 Donald Arseneau 报告的问题，即如果当一段与空白有关的局部代码有效期间，恰巧有一个分页的话，那么在页眉上的标点会出错。这个问题在所有的 TeX 格式中都会出现。举例来说，命令 `\frenchspacing` 和环境 verbatim 就会改变空白的处理方式。

此命令已经在文件的 `\begin{document}` 中自动使用，一般无需显式调用。但也有的类文件中会显式将其定义为其它的非空代码，这样缺省的设置即会被重置。

## 6 升级 LaTeX2.09 的类与宏包

本节主要描述在升级已有的 LaTeX 样式文件为类或宏包时注意哪些问题。当然，也有最理想的情况，即许多已有的样式文件可以不加任何改动地运行在 LaTeX $2_\varepsilon$ 之下，这时，只需在新创建的类或宏包中注明该文档只是为配合新的标准 LaTeX 而发布，然后就可以正常发布给你的用户使用了。

## 6.1 测试能否运行 (Try it first!)

要做的第一件事就是用 '兼容模型' 测试你的样式文件。你所需的唯一修改是更改文件扩展名，如果你的文件是一个主样式文件，则将其扩展名改为 `.cls`。然后不需要作任何其它改动，运行 LaTeX $2_\varepsilon$ 命令对使用了你的文件的文档进行测试。当然，你得有一个文档，其中尽可能多地使用了你在样式文件中提供的新功能 (如果没有这样的文档，你值得花点时间构建一个)。

现在，你需要对测试文件作修改以便使它成为一个 LaTeX $2_\varepsilon$ 的文档，请参阅 *LaTeX $2_\varepsilon$ for Authors* 中的相关内容。然后，对测试文档同时用 LaTeX $2_\varepsilon$ 模式及 LaTeX2.09 兼容模式进行测试。

## 6.2 疑难解答 (Troubleshooting)

如果你的文件不能与 LaTeX $2_\varepsilon$ 配合工作，则可能有两个原因。

- LaTeX 在字体选择方面提供了一种健壮、良好的设计者接口，这与以往的 LaTeX2.09 有较大的差异。
- 样式文件使用了某些已经被更新过了的、或废止了的 LaTeX2.09 内部命令。

在你对这些文件进行调试的过程中，可能希望 LaTeX $2_\varepsilon$ 能提供比标准方式下更多的信息，这可以通过将记数器 `errorcontextlines` 的默认值 $-1$ 调整得更大一些，比如取为 999 等。

## 6.3 使用兼容模式 (Accommodating compatibility mode)

如果已有一系列的 LaTeX2.09 旧文档，要完全抛弃旧的命令可能很不方便，甚至不太可能。这里可以通过一些特殊的命令，使 LaTeX 工作在兼容模式中。

`\if@compatibility`

当一个文件以 `\documentstyle` 而不是 `\documentclass` 开始时，就会设置此开关选项。可以为条件中提供合适的命令序列，例如：

```
\if@compatibility
```

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

## 6.4 字体相关的命令 (Font commands)

在新系统中，一些字体相关的命令是在文档类中定义的，而不在 LaTeX 内核中定义，在将一个 LaTeX2.09 样式文件升级为不装载标准文档类的类文件时，就需要为这些命令提供必要的定义。

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

所有这些简短形式的字体选择命令都不在 LaTeX $2_\varepsilon$ 内核中定义，而是由各种标准文档类文件所定义。

如果需要在你的文档类文件中定义这些命令，合理的方法有好几种。

一种可能的定义为：

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

这样各种字体命令将不会交叉重叠。例如 `{\bf\it text}` 可以产生加粗倾斜效果，即 ***text***。但如果在数学模式中这么使用，则会产生一条错误消息。

另一种可能的定义为：

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

这样定义的 `\rm` 在文本模式下类似于 `\rmfamily` 的行为（见上），同时在数学模式下，又会选择 `\mathrm` 数学字符表。

于是 `${\rm math} = X + 1$` 的效果为 ‘$\mathrm{math} = X + 1$’.

如果没必要字体选择之间不重叠，则可以仿照标准的文档类定义如为：

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

这将意味着 `{\bf\it text}` 会产生中等磅重 (medium weight, 而不是 bold) 的倾斜字体，即 *text*.

`\normalsize`
`\@normalsize`

保留命令 `\@normalsize` 是为了保持 LaTeX2.09 宏包的兼容性，因为这些宏包可能使用了它的取值。但如果在一个文档类中重定义它则无任何效果，因为这样总是将其设置为等价的 `\normalsize` 而已。

因此，文档类必需重定义 `\normalsize` 而不是 `\@normalsize`。例如（不完整）：

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

注意，LaTeX 内核将 `\normalsize` 定义为了一条错误消息。

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

所有这些非 normal 的字体大小命令都没有在内核中定义，如果你需要在一个类文件中使用它们，就必需给出定义。所有的标准文档类中都对它们进行了定义。

可见，在改变字体大小时，对于 `\normalsize` 应当使用 `\renewcommand` 命令，在而对于其它字体大小，则应使用 `\newcommand` 命令。

## 6.5 过时的命令 (Obsolete commands)

有一些宏包不能与 LaTeX 2ε 配合工作，这通常是由于它们依赖于一些 LaTeX 的内部命令，而这些命令在新系统中或者不再支持，或者已经修改过了。

在大多数情形中，都可以使用新系统提供的更高层次的健壮命令，以达到先前低层命令的效果。请参阅第 4 节查看能否用 LaTeX 2ε 的类与宏包命令重写代码。

还有，如果你的宏包内重定义了内核级的命令 (即在 latex.tex, slitex.tex, lfonts.tex, sfonts.tex 等文件中定义的命令)，则应当仔细对照检查这些命令在新内核中的实现情况，特别是检查新内核中是否修改了定义、是否抛弃了该命令等。

由于有太多的 LaTeX2.09 内部命令已经被重新实现或者完全抛弃，这里无法将其一一罗列。你需要加倍仔细地检查你的代码中用到的那些。

这里，我们只列举了一些新系统不再支持的、比较重要的命令。

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

大约有七十余个这样的内部命令已经不再被支持了，如果你的类或宏包中使用了它们，请立即用新的字体命令替代它们（参阅 *LaTeX 2ε Font Selection*）。

例如，命令 \twlsf 应该用以下命令代替：

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

还有一种可能性，就是使用了 rawfonts 宏包，参见 *LaTeX 2ε for Authors* 中的描述。

还有，请记住有许多原本会被 LaTeX2.09 预载的字体在新系统中不会再被预载了。

```
\vpt \vipt \viipt ···
```

这几个是用于改变字体大小的 LaTeX2.09 内部命令 (在 LaTeX2.09 兼容模式下，它们仍可以使用)，请用 \fontsize 命令，细节请参阅 *LaTeX 2ε Font Selection*。

例如，\vpt 应改写为：

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

```
\prm, \pbf, \ppounds, \pLaTeX ···
```

LaTeX2.09 使用了一些以 \p 开头的命令，表示受保护（‘protected’）的意

思。例如，\LaTeX 的定义是 \protect\pLaTeX，其中 \pLaTeX 的定义是产生 LaTeX 的标志（logo）。这样，命令 \LaTeX 是很健壮的，但 \pLaTeX 就不是。

这些命令现在已经用 \DeclareRobustCommand 进行了重新实现（参见第 4.10 节中的描述）。如果在你的宏包中重定义了任何一个 \p- 命令，那么必需移除它们并用 \DeclareRobustCommand 重新定义一个非 \p 打头的命令。

\footheight
\@maxsep
\@dblmaxsep

这些参数在 LaTeX $2_\varepsilon$ 中已经不再使用，因而被完全抛弃了，当然在 LaTeX2.09 兼容模式中仍可用。类文件中不应当再使用它们。

# 参考文献

[1] Michel Goossens, Frank Mittelbach, and Alexander Samarin. *The LaTeX Companion*. Addison-Wesley, Reading, Massachusetts, 1994.

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

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

# LaTeX 2ε 小结：升级旧的样式文件

以下各节号均是指文档 *LaTeX 2ε for Class and Package Writers*。

1. 究竟升级为类还是宏包？第 2.3 节回答了这个问题。
2. 若文件中还用到了另一个样式文件，则需要获取这个样式文件的升级版本，参阅第 2.7.1 节里关于如何装载其它类与宏包的信息。
3. 测试能否运行：见第 6.1 节。
4. 工作正常？太好了，不过也许仍然可以修改以达到健壮性与移植性俱佳、结构良好的 LaTeX 2ε 文件。因此，可参阅第 2 节，特别是第 2.7 小节的内容，另外第 3 节中的例子可能也有帮助。

   如果文档中还有建立新字体、改变字体或符号的命令，还应当参阅第 *LaTeX 2ε Font Selection* 节。
5. 不能正常工作？那么有三种可能性：
   - 读入你的类或宏包文件中报告错误消息；
   - 处理测试文档中报告错误消息；
   - 没有报告错误信息，但输出结果与预期不吻合。

   注意不要忘记仔细地检查这里最后一种情况的出现。

   出现了这些情况，你应该仔细参阅第 6 节的内容以找到合适的解决办法。