

doc 和 shortvrb 宏包*

Frank Mittelbach^{† ‡ §}

2023 年 11 月 1 日 生成

张泓知 翻译

于 2023 年 12 月 10 日

This file is maintained by the L^AT_EX Project team.
Bug reports can be opened (category `latex`) at
<https://latex-project.org/bugs.html>.

摘要

大约 30 年前（版本 1.0 日期为 1988/05/05），我写了 `doc` 包的第一个版本，这是一个用于为 T_EX 代码提供代码文档的包。从那时起，它被广泛用于记录 L^AT_EX 核心和现在大多数可用的包。版本 2 的核心代码（也就是当前版本）自 1998 年存在，也就是已经有 20 年了。

如果我从头开始重新做的话，我会以不同的方式做很多事情，实际上有几个人试图提出更好的解决方案。然而，俗话说，一个不好的标准总比没有要好，`doc` 已经流行了起来，现在以不兼容的方式对其进行更改可能并不真正有益。

因此，这是该包的第 3 版，带有一些较小的扩展，这些扩展是向上兼容的，但希望能够很好地服务。最重要的修改是集成了 `hypdoc` 包的功能，可以在文档内（特别是从索引中）建立链接（如果需要的话）。还集成了 Didier Verna 的 `DoX` 包的想法（尽管我提供了一个与 `doc` 其他接口更好地契合的不同界面）。最后，我更新了一些杂项。

*此文件的版本号为 v3.0m，日期为 2022/11/13。

[†]B. Hamilton Kelly 在 Royal Military College of Science 补充了进一步的评论；Andrew Mills 提供了原始德语评论部分的英文翻译；相当大量的补充内容，特别是来自 `newdoc`，以及 v1.7a 版本中记录了 v1.5q 后功能的文档，由 Dave Love（SERC Daresbury Lab）添加。

[‡]Joachim Schrod（TU Darmstadt）添加了 `shortvrb` 包的提取。

[§]第 3 版现在整合了 Didier Verna 的 `DoX` 包的代码，他的一些文档被重复使用（也可以说是被“偷走”）。

目录

1 介绍	3	7 宏的描述	24
2 用户界面	3	7.1 doc 支持的键	25
2.1 驱动文件	3	7.2 处理包键 (package keys)	25
2.2 包选项	4	7.3 包围“定义部分”的宏	26
2.3 一般约定	5	7.4 “文档部分”的宏	33
2.4 描述宏和环境的用法	6	7.5 格式化边页	35
2.5 描述宏和环境的定义	6	7.6 通过扫描 ‘macrocode’ 创建索引条目	36
2.6 在边页中格式化名称	7	7.7 用于扫描宏名称的宏	38
2.7 提供更多文档条目	7	7.8 索引排除列表	40
2.8 显示示例代码 verbatim	9	7.9 生成索引条目的宏	47
2.9 使用特殊的转义字符	9	7.10 重新定义 index 环境	50
2.10 交叉引用所有使用的宏	9	7.11 处理变更历史	54
2.11 生成实际的索引条目	10	7.12 花哨的功能	57
2.12 设置索引条目	11	7.13 提供校验和字符表	63
2.13 更改默认样式参数的值	12	7.14 将行号附加到代码行	66
2.14 简化输入 verbatim 文 本片段	13	7.15 用于文档化包文件的布 局参数	67
2.15 额外的花哨功能	13	7.16 更改 % 的 \catcode	68
3 示例和基本用法概要	16	7.17 GetFileInfo	68
3.1 基本用法概要	16	8 整合 hypdoc	68
3.2 示例	16	9 集成 DoX 包代码	70
4 版本 2 和版本 3 之间的不兼容性	19	9.1 DoX 环境	70
5 不再真正需要的旧接口	20	9.2 doc 描述	73
5.1 makeindex 的 bug	20	9.3 API 构建	73
5.2 文件传输问题	20	9.4 API 创建	76
6 前版简介	21	9.5 设置默认的 doc 元素	79
		9.5.1 宏设施	79
		9.5.2 环境设施	80
		10 杂项新增	81

1 介绍

这是 doc 包的新版本，大约在初始发布后 30 年左右编写的。由于该包已经被使用了这么长时间（并且基本没有改变），保留现有接口非常重要，即使我们可以同意它们本来可以做得更好。

因此，这只是一个轻量级的更改，基本上添加了超链接支持，并且添加了一种提供一般 doc 元素（不仅限于宏和环境）的方式，并尝试做到这一点（这在过去对于环境也不是这样）。这些想法是从 Didier Verna 的 DoX 包中“借鉴”来的，尽管我没有保留他的接口。

下面大部分的文档来自于之前的版本，这可能导致一些呈现上的不一致，我表示歉意。

2 用户界面

2.1 驱动文件

如果要使用 doc 包来记录一组宏，就必须准备一个特殊的驱动文件，以生成格式化的文档。这个驱动文件具有以下特点：

```
\documentclass[\options]{\document-class}
\usepackage{doc}
\preamble
\begin{document}
\special input commands
\end{document}
```

其中 *\document-class* 可以是任何文档类，我通常使用 `article`。

在 *\preamble* 中，应该放置一些可以控制 doc 包行为的声明，比如 `\DisableCrossrefs` 或 `\OnlyDescription`。

`\DocInput` 最后，*\special input commands* 部分应该包含一个或多个 `\DocInput{<文件名>}` 和/或 `\IndexInput{<文件名>}` 命令。`\DocInput` 命令用于为 doc 包准备的文件，而 `\IndexInput` 则可用于各种类型的宏文件。参见第 14 页，了解 `\IndexInput` 更多细节。可以使用多个 `\DocInput` 命令包含多个文件，每个文件都是独立的、自我包含的、自我记录的包——例如，每个文件都包含 `\maketitle`。

例如，doc 包本身的驱动文件是以下代码，被 `%<*driver>` 和 `%</driver>` 包围。要生成文档，你可以简单地在 L^AT_EX 中运行 `.dtx` 文件，在这种情况下，

这个代码将会被执行（加载文档类 `ltxdoc` 等）；或者，你可以使用 `docstrip` 程序将其提取为一个单独的文件。下面的行号是由 `doc` 的格式化添加的。注意，类 `ltxdoc` 预装载了 `doc` 包。

```
1 <*driver>
2 \documentclass{ltxdoc}
3 % 这里第 3~5 行在英文版的说明文档里是不存在的，出于编译中文的需要，
4 % 被 macrocode 环境以抄录 (verbatim) 的形式显示出来了。
5 \usepackage[fontset=source]{ctex}
6
7 \usepackage[T1]{fontenc}
8 \usepackage{xspace}
9
10 \OnlyDescription
11
12 \EnableCrossrefs
13 % \DisableCrossrefs      % Say \DisableCrossrefs if index is ready
14 \CodelineIndex
15 \RecordChanges          % Gather update information
16 \SetupDoc{reportchangedates}
17 %\OnlyDescription      % comment out for implementation details
18 \setlength\hfuzz{15pt} % don't show so many
19 \hbadness=7000          % over- and underfull box warnings
20 \begin{document}
21   \DocInput{doc-zh-cn.dtx}
22 \end{document}
23 </driver>
```

2.2 包选项

v3 新增

从版本 3 开始，`doc` 包现在提供了一些可以修改其整体行为的包选项。它们包括：

hyperref, **nohyperref** Boolean (default `true`). 载入 `hyperref` 包并使代码行、页码和其他项目的索引引用成为可点击的链接。`nohyperref` 是对应的键。

multicol, **nomulticol** Boolean (default `true`). 用于排版索引和变更列表的 `multicol` 包。`nomulticol` 是对应的键。

debugshow Boolean (default `false`). 在终端和转录文件中提供各种跟踪信息。特别是显示被索引的元素。

noindex Boolean (default `false`). 如果设置, 则抑制所有自动索引。这个选项也可以像下面描述的那样用于单个元素。

noprint Boolean (default `false`). 如果设置, 则抑制边距中元素名称的打印。这个选项也可以像下面描述的那样用于单个元素。

reportchangedates Boolean (default `false`). 如果设置, 则变更条目中在版本号后面列出日期。

\SetupDoc 与向 `doc` 包提供选项不同, 你可以调用 `\SetupDoc` 并在那里提供选项。例如, 这允许在 `doc` 已经加载的情况下更改默认值。

2.3 一般约定

用于与 ‘`doc`’ 包一起使用的 $\text{T}_{\text{E}}\text{X}$ 文件由“文档部分”和“定义部分”交错组成。

“文档部分”的每一行都以百分号 (%) 开头, 位于第一列。它可以包含任意的 $\text{T}_{\text{E}}\text{X}$ 或 $\text{L}^{\text{A}}\text{T}_{\text{E}}\text{X}$ 命令, 但不能使用字符 ‘%’ 作为注释字符。为了允许用户注释, 字符 `^^A` 和 `^^X` 后面都定义为注释字符。¹ 这种“元注释”也可以使用 `\iffalse ... \fi` 来简单地包裹。

文件的其他部分称为“定义部分”。它们包含了“文档部分”中描述的宏的各个部分。

如果文件用于定义新的宏 (例如作为 `\usepackage` 宏中的一个包文件), 高速跳过“文档部分”, 并将宏定义粘贴在一起, 即使它们分为多个“定义部分”也是如此。

macrocode (env.) 另一方面, 如果要生成这些宏的文档, 那么“定义部分”应以抄录 (verbatim) 的形式排版。为了实现这一点, 这些部分应被 `macrocode` 环境包围。更确切地说: 在“定义部分”之前应有一行包含以下内容:

```
%\begin{macrocode}
```

而在这部分之后, 应有一行:

```
%\end{macrocode}
```

在 `%` 和 `\end{macrocode}` 之间必须 恰好 有四个空格—— $\text{T}_{\text{E}}\text{X}$ 在处理“定义部分”时是寻找这个字符串而不是宏。

在“定义部分”内部, 允许使用所有的 $\text{T}_{\text{E}}\text{X}$ 命令; 甚至百分号也可以用于去掉不需要的空格等。

macrocode* (env.) 除了使用 `macrocode` 环境外, 也可以使用 `macrocode*` 环境, 其效果相

¹在版本 2 中只有 `^^A`, 但是许多键盘将 `^` 和 `A` 结合在一起, 并自动转换成 “`Ä`”; 因此, 在版本 3 中添加了 `^^X` 作为备用选项。

同，只是空格被打印为 `\` 字符。

2.4 描述宏和环境的用法

`\DescribeMacro` 当你描述一个新的宏时，可以使用 `\DescribeMacro` 来指示特定宏的使用方式的说明点。它接受一个参数，该参数将打印在页边，并生成特殊的索引条目。例如，我使用 `\DescribeMacro{\DescribeMacro}` 来明确说明这是解释 `\DescribeMacro` 的地方。

由于 `\DescribeMacro` 的参数是一个命令名称，许多人习惯使用（不正确的）简写形式，即在参数周围省略大括号，如 `\DescribeMacro\foo`。只要宏名仅包含“字母”，这种方法是可行的。但是，如果名称包含通常不是“字母”类型的特殊字符（例如 `@`，或者在 `expl3` 中是 `_`、`:`），这将导致严重失败。`\DescribeMacro` 将仅接收到部分命令名（直到第一个“非字母”）例如，`\DescribeMacro\foo@bar` 等同于 `\DescribeMacro{\foo} @bar`，你可以猜到这可能会导致输出不正确，并且可能会出现低级别的错误消息。

`\DescribeEnv` 类似的宏 `\DescribeEnv` 应该用来指示解释一个 `LATEX` 环境。它将生成一个稍微不同的索引条目，并在页边产生略有不同的显示效果。下面我使用了 `\DescribeEnv{verbatim}`。

v3 新增

从版本 3 开始，`\Describe...` 命令接受一个可选参数，您可以在其中指定 `noindex` 或 `noprint`，以抑制该特定实例的索引或打印。同时使用两者也是可能的，但是没有意义，因为此时命令将不再产生任何作用。

2.5 描述宏和环境的定义

`macro (env.)` 为了描述（新）宏的定义，我们使用 `macro` 环境。它有一个参数：新宏的名称。² 此参数也用于在边距中打印名称并生成索引条目。实际上，用于使用和定义的索引条目是不同的，以便轻松引用。此环境可以嵌套。在这种情况下，边距中的标签会垂直排列。在 `\begin{macrocode}` 之前，这个环境里应该有一些文本——即使只是一个空的 `\mbox{}`——否则边距标签将打印在错误的位置。

v3 新增

实际上，现在允许在参数中指定多个宏，用逗号分隔。这是一个简短的形式，用于连续开始多个 `macro` 环境。当然，你也应该只有一个匹配的 `\end{macro}`。

`\MacrocodeTopsep (skip)` 还有四个样式参数：`\MacrocodeTopsep` 和 `\MacroTopsep` 用于控制 `macrocode` 和 `macro` 环境上方和下方的垂直间距。`\MacroIndent` 用于缩进代码行和

`\MacroFont` `\MacroFont` 包含代码行、`verbatim[*]` 环境和边距中打印的宏名称的字

² 这是我在 *TUGboat* 10#1 (1989 年 1 月) 中描述的风格设计变更。我们最终决定最好使用带反斜杠的

体及可能的大小更改命令。如果你想在类文件（如 `ltugboat.cls`）中更改它们的默认值，请使用下面描述的 `\DocstyleParms` 命令。从版本 2.0a 开始，只要重新定义发生在 `\begin{document}` 之前，就可以直接更改它。

`environment (env.)` 为了记录环境的定义，可以使用环境 `environment`，它类似于 `macro` 环境，只是它期望一个 `<env-name>`（不带反斜杠）作为其参数，并在内部提供适合环境的不同索引条目。现在你也可以选择指定一个逗号分隔的环境列表。

v3 新增

从版本 3 开始，这些环境接受一个可选的参数，在其中你可以指定 `noindex` 或 `noprint` 或两者都指定以抑制特定实例的索引或打印。如果在环境级别上进行了这样的设置，它会覆盖在定义 `doc` 元素或加载包时给定的任何默认设置。

2.6 在边页中格式化名称

`\PrintDescribeMacro` 正如前面提到的，一些宏和环境会在边距中打印它们的参数。实际的格式
`\PrintDescribeEnv` 化由四个可用户自定义的宏完成。³ 它们的名称分别是 `\PrintDescribeMacro`
`\PrintMacroName` 和 `\PrintDescribeEnv`（定义了 `\DescribeMacro` 和 `\DescribeEnv` 的行为），
`\PrintEnvName` 以及 `\PrintMacroName` 和 `\PrintEnvName`（分别由 `macro` 和 `environment` 环境调用）。

2.7 提供更多文档条目

v3 新增

初始状态下，`doc` 包提供了上述命令和环境来记录宏和环境。从版本 3 开始，这已经以通用的方式扩展，使你能够轻松提供自己的条目，比如计数器、长度寄存器、选项等。

`\NewDocElement` 提供新的 `doc` 元素的一般语法是：

`\NewDocElement[<options>]{<element-name>}{<env-name>}`

按照惯例，`<element-name>` 的第一个字母是大写的，比如 `Env` 或 `Macro`。

这样的声明将为你定义以下内容：

- 命令 `\Describe<element-name>`，其语法为

`\Describe<element-name>[<options>]{<element>}`

- 环境 `<env-name>`，其语法为

`\begin{<env-name>}[<options>]{<element>}`

宏名称作为参数。

³你可以将更改的定义放在一个单独的包文件中或者放在文档文件的开头。例如，如果你不喜欢边距中的任何名称但想要一个很好的索引，你可以简单地重新定义它们，接受它们的参数但不做任何操作。

- 显示命令 `\PrintDescribe⟨element-name⟩`, 其语法为

`\PrintDescribe⟨element-name⟩{⟨element⟩}`

- 以及环境的显示命令 `\Print⟨element-name⟩Name`。

如果其中任何命令或环境已经被定义（尤其是 `⟨env-name⟩`），这是一个危险的情况），你将收到错误提示。

`\RenewDocElement` 如果你想修改现有的 doc 元素，请使用 `\RenewDocElement`。

例如，已经提供的“Env” doc 元素可以通过简单地声明

`\NewDocElement{Env}{environment}` 来定义，尽管实际情况并非完全如此，稍后我们会看到具体情况。

`⟨options⟩` 是关键字/值对，用于定义 doc 元素的进一步细节。它们包括：

macrolike 布尔值（默认 `false`）。该 doc 元素是否以反斜杠开头？

envlike 布尔值。与 **macrolike** 相对应的选项。

toplevel 布尔值（默认 `true`）。是否应该创建顶级索引条目？如果设置为 `false`，则不会产生任何索引条目，或者只产生分组的索引条目（详见 `idxgroup`）。

notoplevel 布尔值。与 **toplevel** 相对应的选项。

idxtype 字符串（默认 `⟨env-name⟩`）。顶级索引条目的末尾应该放置什么（如果不为空则放在括号内）。

printtype 字符串（默认 `⟨env-name⟩`）。在边距中元素名称后面放置什么（如果不为空则放在括号内）。

idxgroup 字符串（默认 `⟨env-name⟩s`）。如果条目被分组，则顶级索引条目的名称。只有当此选项非空时才进行分组。

noindex 布尔值（默认 `false`）。如果设置，则会抑制此类型元素的索引。此设置会覆盖任何 **noindex** 的全局设置。

noprint 布尔值（默认 `false`）。如果设置，则会抑制在边距中打印元素名称。此设置会覆盖任何 **noprint** 的全局设置。

和往常一样，不带值的布尔选项会将其设置为 `true`。

2.8 显示示例代码 verbatim

`verbatim (env.)` 在文本中包含新宏使用示例通常是个好主意。由于每行首列的 `%` 符号, `verbatim` 环境稍作修改以抑制这些字符。⁴

`verbatim* (env.)` `verbatim*` 环境以相同方式改变。

`\verb` `\verb` 命令被重新实现, 如果其参数中出现换行符则会报错。`verbatim` 和 `verbatim*` 环境将文本设置为 `\MacroFont` 定义的样式 (§2.5)。

2.9 使用特殊的转义字符

`\SpecialEscapechar` 当定义复杂的宏时, 有时需要引入一个新的转义字符, 因为 `'\'` 具有特殊的 `\catcode`。在这种情况下, 可以使用 `\SpecialEscapechar` 来指示实际用于扮演 `'\'` 角色的字符。这样的方案是必要的, 因为 `macrocode` 环境及其对应的 `macrocode*` 会为每个宏名称的出现产生索引条目。如果你不告诉它们你已经改变了 `\catcode`, 它们会非常困惑。`\SpecialEscapechar` 的参数是一个单字母控制序列, 换句话说, 例如要表示 `'\'` 作为转义字符, 就要使用 `\|`。`\SpecialEscapechar` 只会改变接下来的 `macrocode` 或 `macrocode*` 环境的行为。

创建的实际索引条目都会以 `\` 而不是 `|` 打印, 但这可能反映了它们的使用情况, 即使不是它们的定义, 这也比没有任何条目要好。这些条目可能被适当地格式化, 但这样做的效果几乎不值得, 而且生成的索引可能更加混乱(它肯定会更长!)

2.10 交叉引用所有使用的宏

`\DisableCrossrefs` 正如前面提到的, 在 `macrocode` 或 `macrocode*` 环境中使用的每个宏名称都会产生一个索引条目。这样可以很容易地找出特定宏的使用位置。由于当 `TEX` 需要生成如此大量的索引条目时速度会变慢⁵, 我们可以在驱动文件中使用 `\DisableCrossrefs` 关闭此功能。要再次打开此功能, 只需使用 `\EnableCrossrefs`。⁶

`\DoNotIndex` 但也提供了更精细的控制。`\DoNotIndex` 命令接受由逗号分隔的一组宏名称。这些名称不会显示在索引中。你可以使用多个 `\DoNotIndex` 命令: 它

⁴这些宏是由 Rainer Schöpf 编写的 [8]。他还提供了一个新的 `verbatim` 环境, 可在其他宏中使用。

⁵这个评论大约写于 30 年前。虽然现在的 `TEX` 仍然慢得多, 但在处理大型文档 (比如 `LATEX` 内核文档) 时, 过去需要几分钟, 而现在只需要几秒钟甚至更短的时间。因此, 这些天使用 `\DisableCrossrefs` 并不是真的那么必要。

⁶实际上, `\EnableCrossrefs` 更加彻底地改变了事情; 如果源代码中存在后续调用 `\DisableCrossrefs`, 它将被忽略。

们的列表将被串联起来。在本文中，我使用了 `\DoNotIndex` 来排除所有已在 \LaTeX 中定义的宏。

所有上述声明都仅在当前组内有效。

通过在驱动文件的导言部分使用或省略以下声明来控制索引的生成（或不

`\PageIndex` 生成）；如果两者都未使用，则不生成索引。使用 `\PageIndex` 使所有索引条目都引用其页码；使用 `\CodelineIndex`，由 `\DescribeMacro` 和 `\DescribeEnv` 以及可能的进一步 `\Describe...` 命令生成的索引条目引用页码，但由 `macro` 环境（或其他 `doc` 元素环境）生成的索引条目引用代码行数，这些代码行将自动编号。⁷ 此编号的样式可以通过定义宏 `\theCodelineNo` 进行控制。其默认定义是使用 `scriptsize` 的阿拉伯数字；用户提供的定义不会被覆盖。

`\CodelineNumbered` 当你不希望得到一个索引但想要对代码行进行编号时，请使用 `\CodelineNumbered` 而不是 `\CodelineIndex`。这将阻止生成一个不必要的 `.idx` 文件。

2.11 生成实际的索引条目

前面提到的几个宏将产生某种类型的索引条目。这些条目必须由外部程序进行排序——当前的实现假定使用 Chen 编写的 `makeindex` 程序 [4]。

但这并不是内置的：只需重新定义以下一些宏，就能使用任何其他索引程序。所有与安装有关的宏都是以这种方式定义的，以便它们不会覆盖之前的定义。因此，将更改后的版本放入可能在 `doc` 包之前被读取的包文件中是安全的。

为了允许用户更改其索引程序识别的特定字符，所有在 `makeindex` 程序中具有特殊含义的字符都被赋予了符号名称。⁸ 然而，所有使用的字符应该是除了 ‘letter’ (11) 以外的 `\catcode`。

`\actualchar` `\actualchar` 用于分隔“键”和实际的索引条目。
`\quotechar` `\quotechar` 用于在特殊索引程序字符之前，抑制其特殊含义。`\encapchar` 将索引信息与 `makeindex` 用作 \TeX 命令的字母字符串分隔开，用于格式化与
`\levelchar` 特殊条目关联的页码。在此包中，它用于应用 `\main` 和 `\usage` 命令。此外，`\levelchar` 用于分隔“item”、“subitem”和“subsubitem”条目。

即使你知道使用的索引程序，坚持使用这些符号名称也是个好主意。这样你的文件将具有可移植性。

TODO: 描述老的 `\SpecialMainIndex` 和 `\SpecialUsageIndex`

`\SpecialMainMacroIndex` 要为宏生成主索引条目，可以使用 `\SpecialMainMacroIndex` 宏⁹。它被称

`\SpecialMainEnvIndex` ⁷实际上，该行号是 `macro` 环境中第一个 `macrocode` 环境的第一行的行号。

⁸我不知道是否存在需要更多命令字符的程序，但我希望没有。

为“特殊”,因为它必须逐字打印其参数。类似的宏,称为 `\SpecialMainEnvIndex`,用于索引环境的主定义点。¹⁰

`\SpecialMacroIndex` 用于索引宏或环境的使用情况,可以使用 `\SpecialMacroIndex` 和 `\SpecialEnvIndex` `\SpecialEnvIndex`。

v3 新增

所有这些宏通常由其他宏使用;你只在紧急情况下会需要它们。

如果使用 `\NewDocElement{<name>}`... 声明了进一步的代码元素,那么这将设置额外的索引命令,例如, `\SpecialMain<name>Index`。

`\SpecialIndex` `macrocode` 环境会自动索引宏(通常按代码行号)。您也可以(谨慎地)通过 `\SpecialIndex` 手动进行索引。但请注意,如果使用 `\CodelineIndex`,这将生成一个指向最后代码行的条目,通常这不是您想要的。不过,如果您总是只引用页面,即使用 `\PageIndex`,这是有些意义的。

`\SpecialShortIndex` 对于单字符宏,例如 `\{`,并不总是正常工作。因此现在也有了一种特殊变体,可以为它们生成正确的索引条目。

v3 新增

`\SortIndex`

此外,提供了 `\SortIndex` 命令。它接受两个参数——排序关键字和实际索引条目。

`\verbatimchar` 但有一个值得一提的特点:索引中的所有宏名称都使用 `\verb*` 命令排版。因此需要一个特殊字符作为此命令的分隔符。为了允许在这方面进行更改,再次引用了这个字符,即宏 `\verbatimchar`。默认情况下它扩展为 `+`,但如果您的代码行中包含带有 `‘+’` 字符的宏名称(例如当您使用 `\+` 时),这可能会导致问题,因为您最终得到一个包含 `\verb+\++` 的索引条目,它将被排版为 `‘\+’` 而不是 `‘\+’`。在版本 3 中,现在这个问题已经自动处理了(借助 `\SpecialShortIndex` 命令)。

v3 新增

`*` 我们还提供了一个 `*` 宏。它用于这样的索引条目:

```
index entries
Special macros for ~
```

可以使用以下命令生成这样的条目:

```
\index{index entries\levelchar Special macros for \*}
```

2.12 设置索引条目

在第一次通过 `.dtx` 文件进行格式化后,您需要对写入 `.idx` 文件的索引条目进行排序,使用 `makeindex` 或您喜欢的替代工具。您需要一个适合的样式

⁹此宏由 `macro` 环境调用。

¹⁰此宏由 `environment` 环境调用。

文件给 `makeindex` (由 `-s` 开关指定)。在 `doc` 中提供了一个适合的样式文件, 名为 `gind.ist`。

`\PrintIndex` 要读取并打印排序后的索引, 只需将 `\PrintIndex` 命令放在您的包文件中最后一个命令的位置 (作为注释掉的命令, 在文档通过文件时执行)。在其之前, 加上所需的用于引用的任何参考文献命令。或者, 将所有这些调用放置在 `\MaybeStop` 宏的参数之间可能更加方便, 在这种情况下, 您的文件末尾应该出现一个 `\Finale` 命令。

`theindex (env.)` 与标准的 `LATEX` 不同, 默认情况下索引以三列方式排版。这由 `LATEX` 计数器 `IndexColumns (counter)` 控制, 因此可以使用 `\setcounter` 声明进行更改。此外, `\IndexMin (dimen)` 我们不想不必要地开始新页面。因此重新定义了 `theindex` 环境。当 `theindex` 环境启动时, 它将测量当前页面剩余的空间。如果这超过了 `\IndexMin`, 则索引将在此页上开始。否则会调用 `\newpage`。

然后会以单列模式排版一些关于几个索引条目含义的简短介绍。之后, 实际的索引条目以多列模式显示。您可以使用 `\IndexPrologue` 宏更改此序言。实际上, 节标题也是这样生成的, 所以最好写成这样:

```
\IndexPrologue{\section*{Index} The index entries underlined ...}
```

当 `theindex` 环境完成时, 最后一页将重新排版以产生平衡的列。这改善了布局并允许下一篇文章在同一页上开始。索引列的格式 (`\columnsep` 等的值) 由 `\IndexParms` 宏控制。它分配了以下值:

```
\parindent    = 0.0pt          \columnsep    = 15.0pt
\parskip       = 0.0pt plus 1.0pt \rightskip  = 15.0pt
\mathsurround  = 0.0pt          \parfillskip = -15.0pt
```

`\@idxitem` 此外, 它定义了 `\@idxitem` (当遇到 `\item` 命令时将使用) 并选择了 `\small` 大小。如果您想更改其中任何值, 您需要将它们全部重新定义。

`\main` 主索引条目的页码由 `\main` 宏 (对其参数加下划线) 封装, 描述的编号
`\usage` 由 `\usage` 宏封装 (产生斜体)。`\code` 封装在 `macrocode` 环境内解析代码生
`\code` 成的条目中的页码或代码行号。像往常一样, 这些命令是用户可定义的。

2.13 更改默认样式参数的值

`\DocstyleParms` 如果您想覆盖 `doc` 包设置的一些默认设置, 您可以将声明放在驱动文件中 (即在读入 `doc.sty` 后), 或者使用一个单独的包文件来完成这项工作。在后一种情况下, 您可以定义宏 `\DocstyleParms` 包含所有赋值。如果您的包文件在 `doc.sty` 之前读取, 当一些寄存器尚未分配时, 这种间接方法是必要的。其默认定义为空。

目前 doc 包分配值给以下寄存器：

<code>\IndexMin</code>	<code>= 80.0pt</code>	<code>\MacroTopsep</code>	<code>= 7.0pt plus 2.0pt minus 2.0pt</code>
<code>\marginparwidth</code>	<code>= 126.0pt</code>	<code>\MacroIndent</code>	<code>= 17.37196pt</code>
<code>\marginparpush</code>	<code>= 0.0pt</code>	<code>\MacrocodeTopsep</code>	<code>= 3.0pt plus 1.2pt minus 1.0pt</code>
<code>\tolerance</code>	<code>= 1000</code>		

2.14 简化输入 verbatim 文本片段

`\MakeShortVerb` 在文本中引用 verbatim 片段 (例如宏名称) 时, 不断地输入像 `\verb|...|`
`\MakeShortVerb*` 这样的内容很麻烦, 因此提供了一种缩写机制。选择一个字符 $\langle c \rangle$ ——通常其
`\DeleteShortVerb` 类别码为“其他”, 除非您有很好的理由不这样做——并且您不打算在文本
中使用它, 或者不经常使用它。(我喜欢 `"`, 但如果您的 `"` 是活动的用于
umlauts, 您可能更喜欢 `!`。)然后, 如果您使用 `\MakeShortVerb{\langle c \rangle}`, 您随
后可以使用 $\langle c \rangle \langle text \rangle \langle c \rangle$ 作为 `\verb\langle c \rangle \langle text \rangle \langle c \rangle` 的等价形式; 类似地, `*`-形式
的 `\MakeShortVerb*{\langle c \rangle}` 会给您 `\verb*\langle c \rangle \langle text \rangle \langle c \rangle` 的等价形式。如果您
随后希望 $\langle c \rangle$ 恢复到其先前的含义, 则使用 `\DeleteShortVerb{\langle c \rangle}`; 在异
常部分之后, 您总是可以重新开启它。这些“短引用”命令会产生全局更改。
简化的 `\verb` 不能出现在另一个命令的参数中, 就像 `\verb` 一样。但是, “短
引用”字符可以自由地在 verbatim 和 macrocode 环境中使用而不会产生负
面影响。如果 `\DeleteShortVerb` 的参数当前不表示一个短引用字符, 则会被
静默忽略。这两个命令都会显示消息, 告诉您字符的含义正在被更改。

请记住, 命令 `\verb` 不能在其他命令的参数中使用。因此, `\verb` 的缩写
字符也不能在那里使用。

此功能也作为一个单独的包 `shortvrb` 提供。

2.15 额外的花哨功能

我们提供了一些标识的宏, 比如 `WEB`, `AMS-TEX`, `BIBTEX`, `SLITEX` 和
`PLAIN TEX`。只需分别输入 `\Web`, `\AmSTeX`, `\BibTeX`, `\SliTeX` 或 `\PlainTeX`
即可。`LATEX` 和 `TEX` 已在 `latex.tex` 中定义。

`\meta` 另一个有用的宏是 `\meta`, 它有一个参数并生成类似 $\langle dimen parameter \rangle$
的内容。

`\OnlyDescription` 您可以在驱动文件中使用 `\OnlyDescription` 声明来抑制文档的最后部
分 (通常是代码展示部分)。要使其生效, 您需要在文件中的适当位置放置命令
`\MaybeStop`
`\StopEventually` `\MaybeStop`。此宏¹¹ 有一个参数, 在其中放置您希望在文档在此结束时打印

v3 新增

¹¹大约 30 年来, 此宏被称为 `\StopEventually`, 这是由于“假朋友”误解导致的。在德语中, “eventuell”
一词的含义大致是“或许”, 与“eventually”并不完全相同。但鉴于它现在已经使用了这么长时间并且遍布

的所有信息（例如通常在最后打印的参考文献）。如果缺少 `\OnlyDescription` `\Finale` 声明，`\MaybeStop` 宏将其参数保存在一个名为 `\Finale` 的宏中，后续可以使用它来恢复内容（通常在最后）。这样的方案使得不必要地在两个地方进行更改。

因此，您可以使用此功能为 $\text{T}_{\text{E}}\text{X}$ 用户制作本地指南，仅描述宏的使用（大多数用户反正对您的定义不感兴趣）。出于同样的原因，`\maketitle` 命令稍作更改以允许在一个文档中使用多个标题。因此，您可以创建一个驱动文件一次性读取多篇文章。为了避免在标题页上出现意外的 `pagestyle`，`\maketitle` 命令会发出一个 `\thispagestyle{titlepage}` 声明，如果 `titlepage` 页样式未定义，则生成一个 `plain` 页面。这允许像 `ltugboat.cls` 这样的类文件为标题页定义自己的页面样式。

`\AlsoImplementation` 整个文档的排版是默认设置。但是，这个默认设置也可以通过声明 `\AlsoImplementation` 显式选择。这会覆盖任何先前的 `\OnlyDescription` 声明。例如， $\text{L}^{\text{A}}\text{T}_{\text{E}}\text{X} 2_{\epsilon}$ 发行版是使用 `ltxdoc` 类文档化的，它允许使用配置文件 `ltxdoc.cfg`。在这样的文件中，可以添加语句

```
\AtBeginDocument{\AlsoImplementation}
```

来确保所有文档都显示代码部分。

`\IndexInput` 最后但同样重要的是，我定义了一个 `\IndexInput` 宏，它以文件名作为参数，并产生文件的逐行 `verbatim` 列表，同时索引每个命令。如果您想学习一些没有足够文档说明的宏，这可能很方便。我使用这个功能交叉引用了 `latex.tex`，得到了一个大约 15 页索引的逐字稿本。¹²

`\changes` 为了在文件中维护变更历史记录，可以在已更改代码的描述部分中使用 `\changes` 命令。它有三个参数，如下所示：

```
\changes{<version>}{<date>}{<text>}
```

可以使用这些变更来生成一个辅助文件（ $\text{L}^{\text{A}}\text{T}_{\text{E}}\text{X}$ 的 `\glossary` 机制用于此），可以在适当的格式化后打印。`\changes` 宏在这样的变更历史记录中生成打印条目；因为旧版本¹³的 `makeindex` 程序将这些字段限制为 64 个字符，因此在描述变更时应注意不要超过此限制。实际条目由 `<version>`、`\actualchar`、当前宏名称、一个冒号、`\levelchar` 和最后的 `<text>` 组成。结果是一个针对 `<version>` 的词汇条目，其中当前宏的名称作为子项目。在 `macro` 环境之外，文本 `\generalname` 被用作宏名称。在变更描述中引用宏时，通常使用

各地，我们无法放弃旧名称。因此，它仍然存在以允许处理所有现有的文档。

¹²这花了很长时间，生成的 `.idx` 文件比 `.dvi` 文件还长。实际上，太长以至于无法直接由 `makeindex` 程序（在我们的 `MicroVAX` 上）处理，但最终结果值得麻烦。

¹³在 2.6 版之前。

`\cs{\macroname}`}, 而不是尝试正确格式化它并在旧的 `makeindex` 版本中使用宝贵的字符。

注意, 在历史列表中, 条目显示为与其在源文件中的位置相对应的页码, 例如, 放在文件开头的一般变更将显示为页面“1”, 放置在其他位置的变更条目可能具有不同的数字 (不一定总是非常有用, 除非您小心)。

`\RecordChanges` 若要将变更信息写出, 请在驱动文件中包含 `\RecordChanges`。要读取并
`\PrintChanges` 打印排序后的变更历史记录 (以两列形式), 只需将 `\PrintChanges` 命令放在
您的包文件中最后一个命令的位置 (作为注释掉的命令, 在文档通过文件时执行)。或者, 此命令可以形成 `\MaybeStop` 命令的一个参数, 尽管如果只打印描述, 可能不需要变更历史记录。该命令假设 `makeindex` 或其他程序已处理了 `.glo` 文件, 生成了一个排序的 `.gls` 文件。您需要一个特殊的 `makeindex` 样式文件; `doc` 提供了一个合适的样式文件, 名为 `gglo.ist`。

`\GlossaryMin (dimen)` `\GlossaryMin`、`\GlossaryPrologue` 和 `\GlossaryParms` 宏以及计数器
`\GlossaryPrologue` `GlossaryColumns` 与 `\Index...` 版本类似。(L^AT_EX 的“glossary”机制用于变
`\GlossaryParms` 更条目。)

`GlossaryColumns (counter)` 有时需要打印一个 `\`, 但无法使用 `\verb` 命令, 因为符号的 `\catcode` 已
`\bslash` 经被固定了。在这种情况下, 我们可以使用命令 `\bslash`, 前提是此时使用的
实际字体包含“反斜杠”作为符号。请注意, `\bslash` 的定义是可扩展的; 它插入了 `_12`。这意味着如果在“移动参数”中使用它, 您必须对其进行 `\protect` 处理。

`\MakePrivateLetters` 如果您的宏将除 `@` 以外的任何字符设为“字母”, 则应重新定义
`\MakePrivateLetters`, 以便为索引的利益将相关字符也设为“字母”。默认定义只是 `\makeatletter`。

`\DontCheckModules` `docstrip` 系统的“module”指令 [6] 通常被识别并调用特殊格式。可以在
`\CheckModules` `.dtx` 文件或驱动文件中使用 `\CheckModules` 和 `\DontCheckModules` 来开启
`\Module` 或关闭此功能。如果打开了对模块指令的检查 (默认情况下), 则在指令的作用
`\AltMacroFont` 域内的代码将按照 `\AltMacroFont` 钩子决定的格式进行设置。在新字体选择
方案中, 默认情况下, 此钩子给出 *small italic typewriter*; 而在旧的字体选择
方案中, 只是普通的 *small typewriter*, 因为那里不能可移植地使用像斜体打字机这样的字体 (对新字体选择方案的宣传); 如果没有斜体打字机字体可用, 您需要覆盖此设置。代码位于这样的作用域中, 如果它在以 `%<` 开头的行上, 或者在以 `%<*(name list)>` 开头的行和以 `%</ (name list)>` 结尾的行之间。指令由宏 `\Module` 格式化, 其单个参数是尖括号之间 (但不包括) 的指令文本; 此宏可以在驱动或包文件中重新定义, 默认情况下产生类似 `<+foo | bar>` 的结果, 后面没有空格。

`StandardModuleDepth (counter)` 有时 (就像在这个文件中一样), 整个代码被包围在模块中, 以从单个源文

件生成多个文件。在这种情况下，显然不适合将所有代码行都格式化为特殊的 `\AltMacroFont`。出于这个原因，提供了一个计数器 `StandardModuleDepth`，它定义了仍然应该使用 `\MacroFont` 而不是 `\AltMacroFont` 进行格式化的模块嵌套级别。默认设置为 0，对于此文档，在文件的开头它设置为

```
\setcounter{StandardModuleDepth}{1}
```

3 示例和基本用法概要

3.1 基本用法概要

总的来说，没有任何改进的 `.dtx` 文件的基本结构如下：

```
% <waffle>...
...
% \DescribeMacro{fred}
% <description of fred's use>
...
% \MaybeStop{finale code}
...
% \begin{macro}{fred}
% <commentary on macro fred>
%uuuu\begin{macrocode}
<code for macro fred>
%uuuu\end{macrocode}
% \end{macro}
...
% \Finale \PrintIndex \PrintChanges
```

如需进一步了解上述大多数——如果不是全部——特性的使用示例，请参阅 `doc.dtx` 的源代码本身。

3.2 示例

默认设置包括对 doc 元素“宏”和“环境”的定义。它们对应以下声明：

```
\NewDocElement[macrolike = true ,
               idxtype   = ,
               idxgroup  = ,
               printtype =
               ]{Macro}{macro}

\NewDocElement[macrolike = false ,
```



```

idxtype    = env. ,
idxgroup   = environments ,
printtype  = \textit{env.}
]{Env}{environment}

```

为了在某种程度上展示 doc 版本 3 的新功能，当前的文档是通过重新定义这些声明并在顶部添加了一些额外的声明来完成的。

对于我们要记录的任何内部命令，我们使用 Macro 并将所有命令放在“ \LaTeX 命令”标题下（请注意使用 `\actualchar`）。

```

\RenewDocElement[macrolike = true ,
    toplevel   = false,
    idxtype    = ,
    idxgroup   = LaTeX comands\actualchar\LaTeX{} commands ,
    printtype  =
]{Macro}{macro}

```

我们只有包环境，因此对于这些环境，我们使用 Env 并对它们进行分组：

```

\RenewDocElement[macrolike = false ,
    toplevel   = false,
    idxtype    = env. ,
    idxgroup   = Package environments,
    printtype  = \textit{env.}
]{Env}{environment}

```

所有接口命令也被汇总在“包命令”标签下，我们使用 InterfaceMacro 来表示它们：

```

\NewDocElement[macrolike = true ,
    toplevel   = false,
    idxtype    = ,
    idxgroup   = Package commands,
    printtype  =
]{InterfaceMacro}{imacro}

```

由于我们还有一些过时的接口，我们另外添加了一个类别：

```

\NewDocElement[macrolike = true ,
    toplevel   = false,
    idxtype    = ,
    idxgroup   = Package commands (obsolete),
    printtype  =
]{ObsoleteInterfaceMacro}{omacro}

```

另一种类别是包键 (package keys):

```
\NewDocElement[macrolike = false ,  
               toplevel   = false,  
               idxtype    = key   ,  
               idxgroup   = Package keys ,  
               printtype  = \textit{key}  
               ]{Key}{key}
```

最后, 我们有带反斜杠的 $\text{T}_{\text{E}}\text{X}$ 计数器 ($\text{T}_{\text{E}}\text{X}$ counters) 和不带反斜杠的 $\text{L}^{\text{A}}\text{T}_{\text{E}}\text{X}$ 计数器 ($\text{L}^{\text{A}}\text{T}_{\text{E}}\text{X}$ counters), 以及两种类型的 $\text{L}^{\text{A}}\text{T}_{\text{E}}\text{X}$ 长度寄存器 ($\text{L}^{\text{A}}\text{T}_{\text{E}}\text{X}$ length registers):

```
\NewDocElement[macrolike = true ,  
               toplevel   = false,  
               idxtype    = counter ,  
               idxgroup   = TeX counters\actualchar \protect\TeX{} counters ,  
               printtype  = \textit{counter}  
               ]{TeXCounter}{tcounter}
```

```
\NewDocElement[macrolike = false ,  
               toplevel   = false,  
               idxtype    = counter ,  
               idxgroup   = LaTeX counters\actualchar \LaTeX{} counters ,  
               printtype  = \textit{counter}  
               ]{LaTeXCounter}{lcounter}
```

```
\NewDocElement[macrolike = true ,  
               toplevel   = false,  
               idxtype    = skip   ,  
               idxgroup   = LaTeX length\actualchar \LaTeX{} length (skip) ,  
               printtype  = \textit{skip}  
               ]{LaTeXSkip}{lskip}
```

```
\NewDocElement[macrolike = true ,  
               toplevel   = false,  
               idxtype    = dimen  ,  
               idxgroup   = LaTeX length\actualchar \LaTeX{} length (dimen) ,  
               printtype  = \textit{dimen}  
               ]{LaTeXDimen}{ldimen}
```

我们修改索引的外观: 只有两列而不是三列, 并且所有的代码行条目都以

“ ℓ ”（代表行）作为前缀，以便可以轻松地与页面索引条目区分开来。

```
\renewcommand\code[1]{\mbox{$\ell$-#1}}
\renewcommand\main[1]{\underline{\mbox{$\ell$-#1}}}
\setcounter{IndexColumns}{2}
```

4 版本 2 和版本 3 之间的不兼容性

在开发版本 3 时的基本思路是提供与版本 2 非常高的兼容性，以便几乎所有旧文档都可以直接使用，无需进行任何调整。

但是，任何变更都可能导致某种形式的不兼容性，例如，如果新引入的命令名已在用户文档中定义，那么几乎不可能完全避免冲突。

正如之前提到的，doc 现在支持对多个命令和环境进行选项设置，因此，如果“待描述的宏”中使用了像 @ 或 _ 这样的私有字符作为其名称的一部分，则必须在 \DescribeMacro 的参数周围使用大括号。这始终是官方语法，但在过去，你可以更经常地省略这些大括号，而现在则不太容易做到。

旧的 doc 文档还声称可以在加载包之前（不仅之后）重新定义诸如 \PrintDescribeMacro 之类的内容，并且在这种情况下，doc 将不会更改这些命令。由于现在设置机制更加强大和通用，这样的做法实际上并不是很好。所以，在 doc 版本 3 中，修改必须在加载 doc 包之后进行，最后的修改将始终生效。

我曾考虑放弃与 L^AT_EX 2.09 的兼容性（但到目前为止我还保留了它）。

在过去，可以在 \begin{macro} 或 \DoNotIndex 的参数中使用使用 \outer 声明的宏，即使 \outer 不是 L^AT_EX 支持的概念。但这已经不再可能。更确切地说，不再能够阻止它们被索引（因为无法使用 \DoNotIndex），但你可以按以下方式将它们传递给 macro 环境：

```
\begin{macro}[outer]{\foo}
```

如果 \foo 是使用 \outer 声明的宏。这种改变的技术原因是过去，当将诸如 \{ 或 \} 之类的其他命令作为“字符串”而不是单个宏标记传递时，这些参数中的各种命令（如 \{ 或 \}）在处理上并不正确。但是通过切换到宏标记，我们不能再有 \outer 宏，因为它们的特性是不允许出现在参数中。因此，当你使用 [outer] 时，上述操作会将参数读取为四个字符标记的字符串，因此无法识别为 \outer。

5 不再真正需要的旧接口

在计算机程序的生命周期中，三十年是很长的时间，所以在 doc 中有很多接口实际上只是历史性的兴趣（或者当处理同样古老的源文件时才需要）。我们在这里列出它们，但总的来说，我们建议对于新的文档，不要使用它们。

5.1 makeindex 的 bug

`\OldMakeindex` 在 2.9 版本之前的 makeindex 存在一些影响 doc 的 bug。其中一个与 % 字符有关的 bug 在有或没有该 bug 的版本上都没有适当的解决方法。如果你真的还在使用旧版本，可以在包文件或驱动文件中调用 `\OldMakeindex`，以防止索引条目中出现诸如 \% 的问题，尽管通常你可能希望关闭对 \% 的索引。尝试从 T_EX 资源库中获取最新的 makeindex。

5.2 文件传输问题

在互联网早期，文件传输问题曾经是一个严重的问题。在英国罗切斯特有一个著名的网关，处理欧洲大陆到英国的流量，由两台运行不同代码页的 IBM 机器组成（具有不可逆转的差异）。结果是，“奇怪”的 T_EX 字符被替换为其他内容，导致文件变得无法使用。

为了防范这个问题（或者说，如果在传输中出现问题，能够检测到），我在 doc 中添加了代码，用于检查静态字符表，并且还添加了一个非常简单的校验和特性（计数反斜杠）。

如今，`\Checksum` 的价值不大（对开发者来说会很麻烦），字符混淆也不再发生，因此 `\CharacterTable` 实质上已经没有用处。因此，在新的开发中不应该使用它们。

`\CharacterTable` 为了解决在网络上传输文件时出现的一些问题，我们开发了两个宏，可以
`\Checksum` 检测损坏的文件。如果在文档中加入以下内容：

```
%%\CharacterTable
%% {Upper-case   \A\B\C\D\E\F\G\H\I\J\K\L\M\N\O\P\Q\R\S\T\U\V\W\X\Y\Z
%% Lower-case   \a\b\c\d\e\f\g|h|i|j|k|l|m|n|o|p|q|r|s|t|u|v|w|x|y|z
%% Digits       \0\1\2\3\4\5\6\7\8\9
%% Exclamation  \!      Double quote  "      Hash (number) \#
%% Dollar       \$      Percent      \%      Ampersand    \&
%% Acute accent \'      Left paren   \(      Right paren  \)
%% Asterisk     \*      Plus         \+      Comma        \,
%% Minus        \-      Point        \.      Solidus       \/
%% Colon        \:      Semicolon   \;      Less than    \<
%% Equals       \=      Greater than \>      Question mark \?
```

```

%% Commercial at \@      Left bracket \[      Backslash  \\
%% Right bracket \]      Circumflex  \^      Underscore  \_
%% Grave accent  \'      Left brace   \{      Vertical bar |
%% Right brace   \}      Tilde        \~}
%%

```

放置在文件开头，就可以检测到字符翻译失败，前提是所使用的 doc 包具有正确的默认表。每行开头的百分号符号¹⁴ 应该手动输入，因为只有 doc 包应该查看此命令。

邮件发送文件的另一个问题是可能发生截断。为了检测这种类型的错误，我们提供了一个 `\Checksum` 宏。文件的检验和简单地是代码中反斜杠的数量，即位于 `macrocode` 环境之间的所有行。但不用担心：你不必自己计算代码行数；这由 doc 包为您完成。您只需在文件开头附加以下内容：

```
% \Checksum{0}
```

并使用 `\MaybeStop`（它开始查找反斜杠）和 `\Finale` 命令。后者会告诉您，如果您没有检验和（会告诉您正确的数量），或者如果您输入了非零值但猜错了（这次会告诉您正确和错误的数量）。然后，您再次回到文件顶部，将该行更改为正确的数字，即：

```
% \Checksum{<number>}
```

就是这样。

虽然 `\CharacterTable` 和 `\Checksum` 在 doc 写作时的公共互联网早期是重要的特性，因为当时的邮件网关相当不可靠并经常搞乱文件，但在今天，它们更多的是一个麻烦而不是帮助。因此，它们现在是完全可选的，并且不建议在新文件中使用。

6 前版简介

原始摘要：这个包含了格式化包文件文档所需的定义。该包是在 Mainz 与 Royal Military College of Science 合作开发的。这是一个更新，记录了 doc 中的各种

变化和新特性，并集成了 `newdoc` 的功能。

这里描述的 T_EX 宏允许定义和文档保存在同一个文件中。这样做的好处是，通常很复杂的指令通过定义内的注释变得更容易理解。此外，更新

¹⁴每行有两个百分号。这样可以确保这些行不会被 `docstrip.tex` 程序移除。

更容易，只需要更改一个源文件。另一方面，由于这个原因，包文件要长得多：因此 $\text{T}_{\text{E}}\text{X}$ 载入它们需要更长的时间。如果这是一个问题，有一个简单的解决方法：只需要运行 `docstrip.tex` 程序，它会删除几乎所有以百分号开头的行。

集成文档的想法诞生于 $\text{T}_{\text{E}}\text{X}$ 程序的发展；在 Pascal 中用 WEB 系统得以实现。这种方法的优点是显而易见的（可以进行比较 [2]）。自此之后，类似于 WEB 的系统已经为其他编程语言开发出来。但对于最复杂的编程语言之一（ $\text{T}_{\text{E}}\text{X}$ ），文档却被忽视了。在 $\text{T}_{\text{E}}\text{X}$ 世界中似乎存在两种观点：—

- 几个“巫师”，他们写出很多完全难以阅读的代码“即兴演出”，
- 许多用户惊讶于它正是他们所希望的方式工作。或者更确切地说，他们绝望于某些宏拒绝按预期工作。

我不认为 WEB 系统 is 唯一的参

考作品；相反，它是一个原型，足以在 $\text{T}_{\text{E}}\text{X}$ 世界中开发程序。它是足够的，但并非完全足够。¹⁵ 由于 WEB，展示了新的编程视角；不幸的是，对于其他编程语言，这些视角并没有进一步发展。

我在这里介绍的 $\text{T}_{\text{E}}\text{X}$ 宏文档方法也只能被视为一个初步草图。它明确设计为仅在 $\text{L}^{\text{A}}\text{T}_{\text{E}}\text{X}$ 下运行。不是因为我认为这是最好的起点，而是因为从这个起点出发，开发速度最快。¹⁶ 由于这个设计决定，我不得不放弃模块化的概念；这无疑是一步后退。

如果这篇文章能够引发关于 $\text{T}_{\text{E}}\text{X}$ 文档的讨论，我会很高兴。我只能劝告那些认为自己可以在没有文档的情况下应对的人“停止时间”，直到他或她完全理解 $\text{A}_{\text{M}}\text{S}-\text{T}_{\text{E}}\text{X}$ 源代码。

使用 doc 包

与其他任何包一样，在导言部分用 `\usepackage` 命令调用它。由于 doc 使用了 `\reversemarginpars`，可能与一些类不兼容。

版本 1.7 前言（约 1992 年左右）

这个版本的 `doc.dtx` 记录了自上次发布版本 [5] 以来发生的更改，但这些更改已经存在于分发版本的 `doc.sty` 中一段时间了。它还集成了分发的 `newdoc.sty` 中的（未记录的）功能。

自发布版本 [5] 以来，对用户界面进行了以下更改和添加。详细信息请

参见 §2。

驱动机制 现在在驱动文件中使用 `\DocInput` 输入可能是多个独立的 doc 文件，而且 doc 不再必须是最后一个包。`\IndexListing` 被 `\IndexInput` 替换；

索引 由 `\PageIndex` 和 `\CodelineIndex`

¹⁵ 我知道有些人会有不同看法，但这个产品不应该被视为成品，至少就涉及到 $\text{T}_{\text{E}}\text{X}$ 应用方面而言是如此。长期以来关于“多变更文件”的辩论很好地表明了这一点。

¹⁶ 然而，这个论点是错误的，然而，它经常被引用。

控制，其中必须指定一个以生成索引 — 默认的 `\DocstyleParms` 中不再有 `\makeindex`；

macro 环境 现在以带有反斜杠的宏名称作为参数输入；

抄录文本 禁止在 `\verb` 和命令 `\MakeShortVerb`、`\DeleteShortVerb` 内使用换行符；

`\par` 现在可以在 `\DoNotIndex` 中使用；

检验和/字符表支持 用于确保发布的完整性；

`\printindex` 变成了 `\PrintIndex`；

`multicol.sty` 不再是使用 `doc` 或打印文档的必要条件（尽管推荐使用）；

‘Docstrip’ 模块 被识别并特别格式化。

除了添加一些全新的内容外，还在以前在 `newdoc` 中的代码和在版本 1.5k 之后添加的一些注释中添加了一些评论。由于（如相关章节中所述）这些评论不是由 Frank Mittelbach 编写的，而是由代码编写的，因此在这种情况下，“如果代码和评论不一致，那么可能两者都是错误的！”可能是不正确的！

已知问题

这个版本中有一些已知的问题：

- 命令 `\DoNotIndex` 对于一些单字符命令（例如 `\%`）不起作用；
- ‘General changes’ 词汇条目可能会出现在具有前导 `!` 和可能具有前导 `"` 的宏名称之后；
- 如果你有一个旧版本的 `makeindex`，长的 `\changes` 条目会出现奇怪的现象，你可能会发现节标题与第一个更改条目混合在一起。尽量获取一个最新的版本（参见 p. 20）；
- 由于附带的 `makeindex` 样式文件支持旧版和新版 `makeindex` 的不一致属性，在排序索引和更改条目时，`makeindex` 总是会抱怨三个“未知指示符”；
- 如果 `\MakeShortVerb` 和 `\DeleteShortVerb` 与单字符参数一起使用，例如，`{|}` 而不是 `{\|}`，可能会引起混乱。

（某些“功能”在下文有文档记录。）

愿望清单

- 允许 `\DescribeMacro` 和 `\DescribeEnv` 写出关于包的“导出”定义的特殊文件信息的钩子。随后，这可以包含在经过 `docstrip` 的 `.sty` 文件中，以适合于智能编辑器在命令完成、拼写检查等方面的使用，这取决于文档中使用的包。这将需要对“合适形式”达成一致；
- 索引 `docstrip` 的 `%<` 指令中使用的模块；

- 关于包的书目信息的书写；
- 允许关闭特殊字体的使用，例如下一个受保护块。

致谢

我想感谢 Mainz 和皇家军事科学学院的所有人，感谢他们在这个项目中的帮助。特别感谢 Brian 和 Rainer，他们的建议、错误修复等推动了一切。

特别感谢 David Love，在我忽略了这个文件两年多之后，重新更新了文档。这无疑是一项艰巨的工作，因为在 `doc.dtx` 发布在 *TUGboat* 上之后，很多功能从未被正确描述。除了这项出色的工作之外，他还友善地提供了额外的代码（比如“`docstrip`”模块格式化），我相信每个 `doc` 用户都会感激他的贡献。

7 宏的描述

大部分以下的代码经过 `docstrip` 处理后将包含在 `doc.sty` 中，这里指明了要包含的 `style` 模块。（所有不适用于 `doc.sty` 的代码必须显式地由 `docstrip` 包含，以便这个 `.dtx` 文件可以被直接用作包文件，而不是被裁剪版本。）在 `<*style>` 和 `</style>` 指令之间条件包含的代码的通常字体更改被禁止，因为在这个文件中只有带有显式指令的行才是特殊的。

```
24 <*package>
```

在 \LaTeX 2_ϵ 下，避免重复读取 `doc` 通常是不必要的。它仅仅为了保持与 \LaTeX 209 样式的兼容性，因为在那里直接用 `\input` 引入 `doc` 时进行测试是正常无用的。

```
25 \@ifundefined{macro@cnt}{\endinput}
```

`\fileversion` 正如您所看到的，我使用了诸如 `\fileversion` 这样的宏来表示版本号和
`\filedate` 日期。它们在包文件的最开始被定义（没有包裹在 `macrocode` 环境中），所以
`\docdate` 当我改变版本号时，我不必在这里搜索这个地方。您可以在标题的脚注中看到它们的实际输出。

接下来我们要做的第一件事是获取两个可替代的注释符号。由于所有合理的符号都已经被占用了，我们将选择一些只能间接输入的符号：

```
26 \catcode'\^A=14
```

```
27 \catcode'\^X=14
```

我们在文档开头重复这个语句，以防止使用 `inputenc` 宏包将其禁用。

```
28 \AtBeginDocument{\catcode'\^A=14\relax\catcode'\^X=14\relax}
```


7.1 doc 支持的键

在过去, 这使用了 `kvoptions`, 但在未来的某个时候将被 `l3keys` 替代。现在这只是一个轻量级的转变——代码可以和应该进一步修改。

TODO: 清理替换 `kvoptions`

一些键可作为 `\usepackage` 的选项使用, 一些是为生成的项目 API's 提供的:

```
29 \DeclareKeys
30 {
31   noprint          .if    = {doc@noprint},
32   noindex          .if    = {doc@noindex},
33   hyperref         .if    = {doc@hyperref},
34   nohyperref       .ifnot = {doc@hyperref},
35   multicol         .if    = {doc@multicol},
36   nomulticol       .ifnot = {doc@multicol},
37   debugshow       .if    = {doc@debugshow},
38   reportchangedates .if    = {doc@reportchangedates},
39   toplevel         .if    = {doc@toplevel},
40   notoplevel       .ifnot = {doc@toplevel},
41   macrolike        .if    = {doc@macrolike},
42   envlike          .ifnot = {doc@macrolike},
43   idxtype          .store = \doc@idxtype,
44   idxgroup         .store = \doc@idxgroup,
45   printtype        .store = \doc@printtype,
46   outer           .if    = {doc@outer},
47 }
```

最初将这些选项设置为 `true`。

```
48 \doc@hyperreftrue
49 \doc@multicoltrue
50 \doc@topleveltrue
```

7.2 处理包键 (package keys)

```
51 \ProcessKeyOptions
```

`\ifscan@allowed` 当 `\ifscan@allowed` 开启时, 用于确定是否应该启动宏扫描机制的开关, 这个机制是根据 `\active@escape@char` 是否应启动。

```
\scan@allowedtrue
\scan@allowedfalse 52 \newif\ifscan@allowed \scan@allowedtrue
```

`\SetupDoc` 我们需要保存某些选项的默认值, 因为 `doc` 元素可以在局部设置它们。

TODO: 如果可用, 使用 `2e` 接口来设置 `\keys_set:nn`

```

53 \def\SetupDoc#1{%
54   \csname keys_set:nn\endcsname{doc}{#1}%
55   \edef\doc@noprintdefault{\ifdoc@noprint true\else false\fi}%
56   \ifdoc@noindex

```

如果默认情况下我们不进行索引，那么我们也应该关闭 `\scan@allowed`。

```

57   \def\doc@noindexdefault{true}%
58   \scan@allowedfalse
59 \else
60   \def\doc@noindexdefault{false}%
61 \fi
62 }

63 \SetupDoc{} % just save the default values

```

7.3 包围“定义部分”的宏

`macrocode (env.)` 宏定义的部分将被环境 `macrocode` 包围。更准确地说，它们将被一个宏包围，其参数（要设置“抄录”的文本）以字符串 `%\end{macrocode}` 结束。请注意空格的数量。`\macrocode` 完全类似于 `\verbatim` 定义，但由于进行了一些小的改动，几乎所有内部宏都有了新的名称。我们首先调用宏 `\macro@code`，这个宏承担了大部分工作，比如 `\catcode` 重新赋值等。

```
64 \def\macrocode{\macro@code
```

然后我们确保所有空格具有相同的宽度，并且它们不会被丢弃。

```
65   \frenchspacing \@vobeyspaces
```

在结束之前，我们需要调用 `\xmacro@code`。这个宏期望一个由上述字符串终止的参数。这样可以使得 `\catcode` 的改变在局部生效。

```
66   \xmacro@code}
```

`\macro@code` 现在我们将开始实际工作的宏：

```
67 \def\macro@code{%
```

理论上它应该由一个 `trivlist` 环境组成，但是环境前后的空白不应该太大。

```
68   \topsep \MacrocodeTopsep
```

我们接下来设置的参数是 `\@beginparpenalty`，以防止在此类环境之前出现分页。

```
69   \@beginparpenalty \predisplaypenalty
```

然后我们开始一个 `\trivlist`，将 `\parskip` 设置回零，并开始一个空的 `\item`。

```

70   \if@inlabel\leavevmode\fi
71   \trivlist \parskip \z@ \item[]%

```

`\item` 命令设置 `\@labels` 盒子，但该盒子从不排版（通常执行此操作的 `\everypar` 在稍后被重新定义）。通常这不是问题，但在混合方向排版（例如在日语中）时会产生问题，因此我们明确为该用例清除它。

```
72 \global\setbox\@labels\box\voidb@x
```

另外，所有内容应该使用 `typewriter` 字体显示。一些人可能希望有所不同；因此字体选择是由宏驱动的。¹⁷

```
73 \macro@font
```

因为 `\item` 设置了各种参数，我们发现有必要在此之后修改其中一些参数。

```
74 \leftskip\@totalleftmargin \advance\leftskip\MacroIndent
75 \rightskip\z@ \parindent\z@ \parfillskip\@flushglue
```

下一行是 L^AT_EX 中 `\par` 的定义，用于 `\verbatim`，应该导致空白行显示为空白行。

```
76 \blank@linefalse \def\par{\ifblank@line
77 \leavevmode\fi
78 \blank@linetrue\@@par
79 \penalty\interlinepenalty}
```

这个 `\par` 的定义有什么用处呢？我们使用 [3] 的 `\obeylines` 宏，它将所有 `^^M` 改为 `\par`，以便每个 `^^M` 控制其自己的缩进。接下来，我们还必须确保所有特殊符号被归一化；也就是说，它们必须被赋予 `\catcode 12`。

```
80 \obeylines
81 \let\do\do@noligs \verbatim@nolig@list
82 \let\do\@makeother \dospecials
```

如果按代码行进行索引，则增加行号并适当设置。我们还检查下一行的开头是否表示 `docstrip` 模块指令，如果是，则使用 `\check@module` 适当地处理它。

```
83 \global\@newlistfalse
84 \global\@minipagefalse
85 \ifcodeline@index
86 \everypar{\global\advance\c@CodelineNo\@ne
87 \llap{\theCodelineNo\ \hskip\@totalleftmargin}%
88 \check@module}%
89 \else \everypar{\check@module}%
90 \fi
```

我们还通过调用 `\init@crossref` 初始化交叉引用功能。这将在遇到转义字符时启动扫描机制。

```
91 \init@crossref}
```

¹⁷字体更改必须放在 `\item` 之后。否则，对 `\baselineskip` 的更改会影响上面的段落。

`\ifblank@line` `\ifblank@line` 是上面定义中使用的开关。在原始的 `verbatim` 环境中使用了 `\blank@linetrue` `\if@tempswa` 开关。这是危险的，因为在处理 `macrocode` 环境中的行时，其 `\blank@linefalse` 值可能会发生变化。

```
92 \newif\ifblank@line
```

`\endmacrocode` 因为我们在 `macrocode` 环境中开始了一个 `trivlist` 环境，所以我们也必须结束它。我们还必须对 `pm@module` 标志的值进行操作（参见下文）并清空 `\everypar`。

```
93 \def\endmacrocode{%
94         \ifpm@module \endgroup \pm@modulefalse \fi
95         \everypar{}%
96         \global\@inlabelfalse
97         \endtrivlist
```

此外，`\close@crossref` 用于结束交叉引用机制所需的任何操作。

```
98         \close@crossref}
```

`\MacroFont` 这里是 `\MacroFont` 宏的默认定义。随着 NFSS2 中新的数学字体处理方式的出现，抑制数学字体设置不再正确，因为现在处理方式不同了。但为了保持字体更改的快速性，我们只使用一个 `\selectfont`（在 `\small` 中），其余部分手动处理。

```
99 \@ifundefined{MacroFont}{%
100   \if@compatibility
```

尽管上面的声明，如果有人正在使用带有 `doc` 的 L^AT_EX2.09 文档，我们将首先调用 `\small`。我本来不会费这个劲的，因为 `doc` 源文件应该是最新的，但由于这个请求来自一个叫 David Carlisle 的人 ...:-)

```
101   \def\MacroFont{\small
102           \usefont\encodingdefault
103           \ttdefault
104           \mddefault
105           \shapedefault
106           }%
107   \else
108   \def\MacroFont{\fontencoding\encodingdefault
109           \fontfamily\ttdefault
110           \fontseries\mddefault
111           \fontshape\shapedefault
112           \small}%
113   \fi
114   }
```

`\AltMacroFont` 虽然大部分宏代码都设置在 `\MacroFont` 中，但我们希望能够切换到指示在 `\macro@font` `\AltMacroFont` 中设置的模块代码。`\macro@font` 跟踪我们正在使用的字体。在 OFSS 中我们不能像在 NFSS 中那样明智地做同样的事情。

```
115 \@ifundefined{AltMacroFont}{%
116   \ifcompatibility
```

如果我们处于兼容模式下，再次首先使用 `\small`。

```
117   \def\AltMacroFont{\small
118                               \usefont\encodingdefault
119                               \ttdefault
120                               \mddefault
121                               \sldefault
122                               }%
123   \else
124   \def\AltMacroFont{\fontencoding\encodingdefault
125                               \fontfamily\ttdefault
126                               \fontseries\mddefault
127                               \fontshape\sldefault
128                               \small
129                               }%
130 \fi
131 }
```

为了允许在导言部分更改 `\MacroFont`，我们将内部使用的 `\macro@font` 的定义推迟到导言部分之后。

```
132 \AtBeginDocument{\let\macro@font\MacroFont}
```

`\check@module` 这个宏会在每个宏代码行的开头由 `\everypar` 插入，用于检查该行是否以模块信息开头。（此类信息的格式为 `%< 开关>`，其中 `%` 必须位于行首，`< 开关>` 由各种可能的分隔符和可能的前导 `+`、`-`、`*` 或 `/` 组成 [6]。这里我们关心的是 `< 开关>` 的第一个字符。）首先，它检查 `pm@module` 标志，以防前一行有一个非块模块指令，即不是 `%<*` 或 `%</`；如果是，我们需要关闭它开始的组并取消该标志。`\check@module` 向前查看下一个记号，然后调用 `\ch@percent` 根据它是否是 `%` 来采取相应的操作；我们在这个阶段不想展开记号。所有这些都是有条件地进行的，因此如果它引起没有设计为进行 `docstrip` 的代码出现问题，可以将其关闭。

```
133 \def\check@module{%
134   \ifcheck@modules
135     \ifpm@module \endgroup \pm@modulefalse \fi
136     \expandafter\futurelet\expandafter\next\expandafter\ch@percent
137   \fi}
```

```
138 \newif\ifpm@module
```

`\DontCheckModules` 这里有两个驱动文件接口宏，用于使用 `check@modules` 开关打开和关闭模块
`\CheckModules` 检查。

```
\ifcheck@modules 139 \def\DontCheckModules{\check@modulesfalse}
140 \def\CheckModules{\check@modulestrue}
141 \newif\ifcheck@modules \check@modulestrue
```

`\ch@percent` 如果 `\next` 中的前瞻记号是 $\%_{12}$ ，我们继续检查接下来的记号是否是 `<`，否则什么也不做。注意使用 `\expandafter` 来跳过 `\fi`。

```
142 \def\ch@percent{%
143   \if \percentchar\next
144     \expandafter\check@angle
145   \fi}
```

`\check@angle` 在寻找 `<` 之前，这里的参数会吞掉 `%`。

```
146 \def\check@angle#1{\futurelet\next\ch@angle}
```

`\ch@angle` 如果当前的前瞻记号是 `<`，我们就定义为正在处理模块指令，并继续寻找 `+` 等；否则，我们必须将吞掉的 `%` 放回去。在 $\text{\LaTeX 2}_{\epsilon}$ 中，`<` 是活动字符，所以我们必须小心一些。

```
147 \begingroup
148 \catcode'\<\active
149 \gdef\ch@angle{\ifx<\next
150   \expandafter\ch@plus@etc
151   \else \percentchar \fi}
```

`\ch@plus@etc` 现在我们必须决定我们正在处理的是什么类型的指令，并对其进行正确处理。

```
\check@plus@etc 152 \gdef\ch@plus@etc<{\futurelet\next\check@plus@etc}
153 \gdef\check@plus@etc{%
154   \if +\next
155     \let\next\pm@module
156   \else\if -\next
157     \let\next\pm@module
158   \else\if *\next
159     \let\next\star@module
160   \else\if /\next
161     \let\next\slash@module
```

在过去的某个时候，`docstrip` 程序部分重写，并在那时也获得了对形如 `%<<` 后跟任意字符串的非常特殊指令的支持。这用于在出现某些问题时进行“逐字”包含。我们实际上并不尝试对这种情况进行美化，但至少我们需要考虑到

它，因为否则我们会得到一个错误消息，因为这是唯一一个我们不会有结束 > 的情况。

```

162   \else\ifx <\next
163     \percentchar
164   \else
165     \let\next\pm@module
166   \fi\fi\fi\fi\fi
167   \next}
168 \endgroup

```

`\pm@module` 如果我们不处理一个块指令 (* 或 /)，即它是一行特殊行，我们会将一切设置为下一个 > 并在组内更改到特殊的宏字体，该组将在下一行的开始时结束。如果表面上的模块指令缺少终止的 >，这将丢失，但 `docstrip` 实现也会如此。另一种策略是让 `\pm@module` 使 > 成为活动字符，并清除此处设置的标志，以指示正在处理该指令。如果在处理下一行时发现该标志仍然设置，则可以采取适当的操作。

```

169 \begingroup
170 \catcode'\~=\active
171 \lccode'\~='>
172 \lowercase{\gdef\pm@module#1~}{\pm@moduletrue
173   \Module{#1}\begingroup

```

当嵌套大于 `\c@StandardModuleDepth` 的当前值时，我们立即切换到特殊字体。我们在这里对 `\guard@level` 进行局部更新，在当前输入行之后将其恢复。

```

174   \advance\guard@level\@ne
175   \ifnum\guard@level>\c@StandardModuleDepth\AltMacroFont\fi
176 }

```

`\star@module` 如果指示了模块的开始或结束块，在设置守卫之后，我们必须检查是否应该更改宏代码字体。如果我们已经在块内部或者正在结束最外层的块，则会出现这种情况。如果是这样，我们会在后续的宏代码部分全局切换普通形式和特殊形式的字体，并立即切换到新的字体。

```

177 \lowercase{\gdef\star@module#1~}{%
178   \Module{#1}%
179   \global \advance \guard@level\@ne
180   \ifnum \guard@level>\c@StandardModuleDepth
181     \global\let\macro@font=\AltMacroFont \macro@font
182   \fi}
183 \catcode'\>=\active
184 \gdef\slash@module#1>{%
185   \Module{#1}%

```

```

186 \global \advance \guard@level\m@ne
187 \ifnum \guard@level=\c@StandardModuleDepth
188 \global\let\macro@font\MacroFont \macro@font
189 \fi
190 }
191 \endgroup

```

`StandardModuleDepth` (*counter*) 计数器定义了哪个级别的模块被视为主代码的一部分。例如，如果整个代码被 `%<*package>` 模块包围，最好将此计数器设置为 1，以避免整个代码显示为等宽斜体。

```

192 \newcounter{StandardModuleDepth}

```

`\guard@level` (*counter*) 我们需要一个计数器来跟踪守卫的嵌套。

```

193 \newcount \guard@level

```

`\Module` 这提供了一个钩子来确定模块指令的设置方式。它以尖括号之间的所有内容作为参数。默认情况下，使用无衬线字体在 $\langle \rangle$ 之间设置内容，并使用 `\mod@math@codes` 适当地 `\mathcode` 特殊字符。（你不能只是在无衬线文本字体中设置它，因为通常 `|` 会打印为破折号。）这取决于我们使用的是 NFSS 还是旧版。在后者的情况下，我们可以轻松地相应地更改 `\fam`。

```

194 \@ifundefined{Module}{%
195 \def\Module#1{\mod@math@codes$\langle\mathsf{#1}\rangle$}
196 }{}

```

`\mod@math@codes` 除了 ‘words’ 外，模块指令文本可能包含任何字符 `*/+-,&|!()` 对于当前版本的 docstrip。在上面所需的数学代码更改中，我们只需要对其中的两个进行特殊处理：`|` 被更改为 `\mathop`（通常为 "026A），`&` 也被变为一个 `\mathop`，但在族 0 中。请记住，在遇到 `&` 时，它不会具有特殊的类别码。

```

197 \def\mod@math@codes{\mathcode'\|= "226A \mathcode'\&="2026
198 \mathcode'\-= "702D \mathcode'\+= "702B
199 \mathcode'\:= "703A \mathcode'\== "703D }

```

`\MacrocodeTopsep` (*skip*) 在上面的代码中，我们使用了两个寄存器。因此，我们必须分配它们。默认值 `\MacroIndent` (*dimen*) 可以通过 `\DocstyleParms` 宏进行覆盖。

```

200 \newskip\MacrocodeTopsep \MacrocodeTopsep = 3pt plus 1.2pt minus 1pt
201 \newdimen\MacroIndent
202 \settowidth\MacroIndent{\rmfamily\scriptsize 00\ }

```

`macrocode*` (*env.*) 就像 `verbatim` 环境一样，`macrocode` 环境也有一个 ‘star’ 变种，其中空格由 `\endmacrocode*` 符号 `□` 显示。到目前为止，我还没有使用它（它将在下面对 `\xmacro@code` 的

定义描述中使用)，但恰好在这一次这里 你不能直接使用它（参见蒙特豪森的沼泽问题）¹⁸。因此，在这个特殊情况下，我们将通过额外的注释字符绕过这个问题。但现在回到 `\macrocode*`。我们从准备一切的宏 `\macro@code` 开始，然后调用宏 `\sxmacro@code`，其参数以字符串 `%\end{macrocode}` 结束。

```
203 \namedef{macrocode*}{\macro@code\sxmacro@code}
```

如我们所知，将执行 `\sxmacro@code` 然后执行 `\end{macrocode}`（宏，而不是字符串），所以为了圆满结束，我们仍然需要定义宏 `\endmacrocode*`。

```
204 \expandafter\let\csname endmacrocode*\endcsname = \endmacrocode
```

`\xmacro@code` 正如前面提到的，宏 `\xmacro@code` 需要由字符串 `%\end{macrocode}` 分隔的参数。在调用此宏时， \TeX 的特殊字符的 `\catcode` 为 12（“其他”）或 13（“活动”）。因此，在定义时我们需要使用不同的转义字符。这是在本地进行的。

```
205 \begingroup
```

```
206 \catcode'\|=\z@\catcode'\[=\@ne\catcode'\]=\tw@
```

此外，我们需要确保上述字符串中包含 `macrocode` 环境内可用的 `\catcode`。

```
207 \catcode'\{=12\catcode'\}=12
```

```
208 \catcode'\%=12\catcode'\_=\active\catcode'\\=\active
```

接下来是 `\macro@code` 的实际定义；请注意新转义字符的使用。我们设法让参数被字符串 `\end{macrocode}` 包围，但是最后，尽管在此宏的定义过程中使用了实际字符，但是 `\end` 与参数 `{macrocode}` 将会执行，以确保环境平衡。

```
209 \gdef\xmacro@code#1%\end{macrocode}[#1\end{macrocode}]
```

`\sxmacro@code` `\sxmacro@code` 的定义完全类似，只是此处将使用稍微不同的终止字符串。请注意，此环境中空格不是活动字符。

```
210 \catcode' =12
```

```
211 \gdef\sxmacro@code#1% \end{macrocode*}[#1\end{macrocode*}]
```

因为 `\catcode` 的更改是通过启动新组而在本地进行的，所以现在以一种非常不寻常的写法跟随着匹配的 `\endgroup`。

```
212 \endgroup
```

7.4 “文档部分”的宏

为了将标签放在左边栏，我们必须使用 `\reversemarginpar` 声明。（这意味着 `doc.sty` 无法与所有类或包一起使用。）我们还将 `\marginparpush` 设为

¹⁸Karl Friedrich Hieronymus Frhr. v. Münchhausen (*1720, †1797)。有几本书记载了据说是他讲述的奇幻冒险（参见 [7] 或 [1]）。在其中一则故事中，他通过用自己的头发拉自己爬出了沼泽。

零, 并将 `\marginparwidth` 设为足够宽。

```
213 \reversemarginpar
214 \setlength\marginparpush{0pt} \setlength\marginparwidth{8pc}
215 \setlength\marginparsep{\labelsep}
```

`\bslash` 我们在一个新的组中隐藏 `\catcode` 的修改, 并让 `|` 引入命令, 而 `\` 成为一个“其他”字符。

```
216 {\catcode'\|=z@ \catcode'\=12
```

现在我们可以定义 `\bslash` (全局地) 生成一个 `\catcode` 为“其他”的反斜杠。然后我们关闭这个组, 恢复原始的 `\catcode`。

```
217 |gdef|bslash{\}}
```

`verbatim (env.)` `verbatim` 环境没有秘密; 它由正常的 `LATEX` 环境组成。我们还设置了 `\@beginparpenalty` `verbatim* (env.)` 并改变了 `\MacroFont` 所给定的字体。

```
218 \def\verbatim{\@beginparpenalty \predisplaypenalty \@verbatim
219 \MacroFont \frenchspacing \vobeyspaces \@xverbatim}
```

我们以类似的方式处理此环境的星号形式。

```
220 \@namedef{verbatim*}{\@beginparpenalty \predisplaypenalty \@verbatim
221 \@setupverbvisiblespace
222 \MacroFont \vobeyspaces \@sxverbatim}
```

`\@verbatim` 此外, 我们重新定义了 `\@verbatim` 宏, 以便抑制行首的 `%` 字符。前几行直接从 `latex.tex` 中复制过来。

```
223 \def\@verbatim{\trivlist \item\relax
224 \if@minipage\else\vskip\parskip\fi
225 \leftskip\@totalleftmargin\rightskip\z@
226 \parindent\z@\parfillskip\@flushglue\parskip\z@
227 \language\l@nohyphenation
228 \@@par
229 \@tempwafalse
```

`\@verbatim` 将 `^M` (行尾字符) 设置为等于 `\par`。此控制序列在此重新定义; `\@@par` 是 `TEX` 的段落原语。

```
230 \def\par{%
231 \if@tempwa
232 \leavevmode \null \@@par\penalty\interlinepenalty
233 \else
234 \@tempwattrue
235 \ifhmode\@@par\penalty\interlinepenalty\fi
236 \fi
```

我们在`\par`的定义中添加了一个控制序列`\check@percent`，其任务是检查百分号字符。

```
237 \check@percent}%
```

剩下的部分再次直接从 `latex.tex` 中复制（少了`"`）。

```
238 \let\do\@makeother \dospecials
239 \obeylines \verbatim@font \noligs
240 \everypar \expandafter{\the\everypar \unpenalty}%
241 }
```

`\check@percent` 最后我们定义了`\check@percent`。因为这需要比较一个字符和百分号，我们必须首先（局部地）改变百分号的`\catcode` 以便 `TEX` 可以看到它。定义本身几乎是微不足道的：获取下一个字符，检查它是否是 `%`，如果不是，则再次插入。在 `verbatim` 环境的末尾，此宏将窥视下一个输入行。在这种情况下，`\check@percent` 的参数可能是一个`\par` 或具有参数的宏。因此，我们将定义为 `\long`（允许`\par`），并在需要使用正常的`\next` 机制在`\fi` 之后重新插入参数。这里有一个微妙的问题，`\next` 和 `#1` 之间的等号实际上是必需的。你明白为什么吗？一次错误地省略了这个标记引起了一个有趣的错误。

```
242 {\catcode'\%=12
243 \long\gdef\check@percent#1{\ifx #1%\let\next\@empty \else
244 \let\next=#1\fi \next}}
```

在该宏包的早期版本中，它还重新定义了`\verb`，因为它没有在抄录文本中包括对“换行符”的有用测试。这在现在的 `LATEX` 中已经包含，因此我们不再重新定义它（原始代码仍然保留在`\endinput` 后的文件中，以保持完整的历史记录）。

`\macro@cnt (counter)` `macro` 环境实现为 `trivlist` 环境，为了使宏名称能够在边距下放置在一起（对应于宏的嵌套深度），必须改变宏`\makelabel`。为了存储嵌套深度，我们使用一个计数器。我们还需要一个计数器来计算嵌套的 `macro` 环境的数量。

```
245 \newcount\macro@cnt \macro@cnt=0
```

`\MacroTopsep (skip)` 这里是上面使用的 `\MacroTopsep` 参数的默认值。

```
246 \newskip\MacroTopsep \MacroTopsep = 7pt plus 2pt minus 2pt
```

7.5 格式化边页

下面的三个宏应该是用户可定义的。因此，我们只在它们尚未被定义时定义这些宏。

7.6 通过扫描 ‘macrocode’ 创建索引条目

下面的宏确保为每个类似于 `TEX` 命令（以 ‘\’ 开头的内容）创建索引条目，前提是使用了 `\PageIndex` 或 `\CodelineIndex` 打开了索引。对于 `\specialMainMacroIndex` 等的默认定义，所生成的索引文件预期要由 Chen 的 `makeindex` 程序 [4] 处理。

当然，在此宏包文件中，我们有时不得不让 | 扮演 `TEX` 的转义字符的角色，以在 \ 必须属于其他类别的地方引入命令名。因此，在设置 `macrocode` 环境内的文本时，我们可能还需要将 | 识别为命令的引入符号。其他用户可能也需要对其宏进行类似的重新分配。

`\SpecialEscapechar` 宏 `\SpecialEscapechar` 用于表示下一个 `macrocode` 环境的特殊转义字符。它有一个参数——作为“单字母”控制序列给出的新转义字符。它的主要目的是定义 `\special@escape@char` 以产生选定的转义字符的 `\catcode` 为 12，并定义 `\active@escape@char` 以产生相同字符，但 `\catcode` 为 13。

宏 `\special@escape@char` 用于打印转义字符，而 `\active@escape@char` 则在 `\init@crossref` 的定义中需要启动扫描机制。

在定义 `\SpecialEscapechar` 中，我们需要一个带有 `\catcode` 13 的任意字符。我们使用了 ‘~’ 并确保其是活动字符。`\begingroup` 用于使可能的更改局部化到 `\SpecialEscapechar` 的展开中。

```
247 \begingroup
248 \catcode'\~\active
249 \gdef\SpecialEscapechar#1{%
250     \begingroup
```

现在我们准备定义 `\active@escape@char`。这有点棘手：我们首先在本地定义 ‘~’ 的大写代码为新的转义字符。

```
251     \uccode'\~'#1%
```

在 `\active@escape@char` 的定义周围，我们放置了一个 `\uppercase` 命令。请记住，`\uppercase` 的展开根据其 `\uccode` 改变字符，但保持其 `\catcode` 不变（参见 `TEXbook` 第 41 页）。

```
252     \uppercase{\gdef\active@escape@char{~}}%
```

定义 `\special@escape@char` 更简单，我们使用 `\string` 将 `\SpecialEscapechar` 的参数的 `\catcode` 改为 12 并抑制前置的 `\escapechar`。

```
253     \escapechar\m@ne \xdef\special@escape@char{\string#1}%
```

现在我们关闭组并结束定义：`\escapechar` 的值以及 ‘~’ 的 `\uccode` 和 `\catcode` 将被恢复。

```
254 \endgroup}
255 \endgroup
```

`\init@crossref` `\init@crossref` 的替换文本应执行以下任务：

- 1) 将所有宏名称中使用的字符的 `\catcode` 设为 11 (即 ‘letter’)
- 2) 将 ‘\’ 字符的 `\catcode` 设为 13 (即 ‘active’)
- 3a) 如果没有特殊转义字符 (即 `\special@escape@char` 为 ‘\’), 则将 ‘\’ 等于 `\scan@macro` (即开始宏扫描机制)。
- 3b) 否则将其等于 `\bslash`, 即产生可打印的 ‘\’。
- 4) 使 *⟨special escape character⟩* 成为活动字符。
- 5) 将特殊转义字符的活动版本 (即 `\active@escape@char` 的扩展) 设为 `\scan@macro`。

读者可能会问为什么我们首先将 ‘\’ 的 `\catcode` 设为 12 (在 `\macro@code` 结尾处), 然后在上面的 3b) 中重新将其设为 13, 以产生一个 `_12`。这是因为我们必须确保在 `macrocode` 环境内 ‘\’ 的 `\catcode` 为 13。否则, 对于 `\xmacro@code` 的参数匹配, 不会找到分隔符 (参数匹配取决于 `\catcode`)。

因此, 我们首先重新设定一些字符的 `\catcode`。

```
256 \begingroup \catcode'\!=\z0 \catcode'\!=\active
```

我们首先执行 2) 和 3b) 的任务。

```
257 |gdef\init@crossref{\catcode'\!=\active |let\bslash
```

由于 ‘@’ 字符在宏中作为 ‘letter’ 的普及性, 我们通常需要在此处更改其 `\catcode`, 从而实现任务 1)。但是宏设计者可能还使用其他字符作为私有字母, 因此我们使用一个宏来进行 `\catcode` 切换。

```
258 |MakePrivateLetters
```

现在我们将特殊转义字符的 `\catcode` 设为 13, 并将其 `\let` 等于 `\scan@macro`, 即完成了任务 4) 和 5)。请注意使用 `\expandafter` 插入保存在 `\special@escape@char` 和 `\active@escape@char` 中的选择的转义字符。

```
259 |catcode\expandafter\special@escape@char\active
```

```
260 |expandafter\let\active@escape@char\scan@macro}
```

```
261 |endgroup
```

如果没有特殊转义字符, 即如果 `\SpecialEscapechar` 是 `\`, 倒数第二行将覆盖之前的定义 `_13`。通过这种方式, 所有任务都得以完成。

为了便于文档编写, 我们为 `\special@escape@char` 和 `\active@escape@char` 给出了默认值:

```
262 \SpecialEscapechar{\}
```

`\MakePrivateLetters` 这是此命令的默认定义，只将 `@` 转换为字母。用户可以根据需要更改它，以使更多或其他字符伪装成字母。

```
263 \ifundefined{MakePrivateLetters}
264     {\let\MakePrivateLetters\makeatletter}{}
```

`\close@crossref` 在交叉引用部分结束时，我们通过将转义字符设置为 ‘\’ 来准备下一个部分。

```
265 \def\close@crossref{\SpecialEscapechar\}
```

7.7 用于扫描宏名称的宏

`\scan@macro` `\init@crossref` 将会使得 `\special@escape@char` 变为 `\active`，因此每个 `\macro@namepart` `\active@escape@char` 在 `macrocode` 环境内会调用 `\scan@macro`。通过这种方式，我们可以在设置（在 `verbatim` 中）`macrocode` 环境的内容时自动为每个类似 `TEX` 命令添加索引条目。

```
266 \def\scan@macro{%
```

首先输出触发此宏的字符。其被 `\catcode` 设为 12 的版本保存在 `\special@escape@char` 中。我们还调用 `\step@checksum`，以便稍后生成正确的校验和（有关详细信息，请参阅第 5.2 节）。

```
267     \special@escape@char
268     \step@checksum
```

如果 `macrocode` 环境中包含，例如，命令 `\\`，第二个 `\` 不应启动扫描机制。因此，我们使用开关来决定是否允许扫描宏名称。

```
269     \ifscan@allowed
```

该宏将组合形成 `TEX` 命令的字母放入 `\macro@namepart` 中，因此初始情况下清除它；然后将 `\next` 设置为 `\` 后面的第一个字符，并调用 `\macro@switch` 来确定该字符是否是字母。

```
270         \let\macro@namepart\@empty
271         \def\next{\futurelet\next\macro@switch}%
```

正如你所看到的，实际上我们做了一些其他的事情，因为我们必须推迟 `\futurelet` 调用，直到最后的 `\fi` 之后。另一方面，如果禁用了扫描，则我们简单地将 `\next` 设为 ‘empty’。

```
272     \else \let\next\@empty \fi
```

现在我们调用 `\next` 执行所需的操作。

```
273     \next}
```

`\EnableCrossrefs` 此时我们可以定义两个宏，允许用户禁用或启用交叉引用机制。如果仅生成主索引条目（即，如果启用了 `\DisableCrossrefs`），则文件处理速度会更快。

```
274 \def\DisableCrossrefs{\@bsphack\scan@allowedfalse\@esphack}
```

宏 `\EnableCrossrefs` 也会禁用之后遇到的任何 `\DisableCrossrefs` 命令。

```
275 \def\EnableCrossrefs{\@bsphack\scan@allowedtrue
```

```
276 \def\DisableCrossrefs{\@bsphack\@esphack}\@esphack}
```

`\macro@switch` 现在我们已经获取了跟在转义字符后面的字符（存储在 `\next` 中），我们可以确定它是否是一个“字母”（可能包括 `@`）。

如果是，我们让 `\next` 调用一个宏来组装完整的命令名称。

```
277 \def\macro@switch{\ifcat\noexpand\next a%
```

```
278 \let\next\macro@name
```

否则，我们有一个“单字符”命令名称。对于所有这些单字符名称，我们使用 `\short@macro` 将其处理成合适的索引条目。

```
279 \else \let\next\short@macro \fi
```

现在我们知道要使用哪个宏来处理宏名称，我们调用它 ...

```
280 \next}
```

`\short@macro` 当在 `macrocode` 环境内扫描到单字符宏名称时，此宏将被调用（带有一个单字符作为参数）。

首先，我们查看 `\index@excludelist`，看看这个宏名称是否应该产生索引条目。这由 `\ifnot@excluded` 宏完成，假设宏名称保存在 `\macro@namepart` 中。字符不能以特殊的类别码存储，否则将无法排除索引，所以我们使用 `\string` 来规范化它，就像在 `\DoNotIndex` 中做的那样，即除了空格字符外，其他所有字符都以类别码 12 结束。

```
281 \def\short@macro#1{%
```

```
282 \edef\macro@namepart{\string#1}%
```

然后任何索引工作都委托给 `\maybe@index@short@macro`。根据实际字符的不同，此宏必须执行不同的操作，因此我们将其与 `\maybe@index@macro` 分开，以避免在多字母宏名称的常见情况下进行特殊测试。

```
283 \maybe@index@short@macro\macro@namepart
```

接着我们使用 `\scan@allowedfalse` 禁用交叉引用机制，并打印实际字符。首先生成索引条目以确保不会出现分页（请注意，`^^M` 将开始新行）。

```
284 \scan@allowedfalse#1%
```

在排版字符之后，我们可以安全地再次启用交叉引用功能。请注意，如果全局禁用交叉引用，则不会调用此宏（因为不会调用 `\macro@switch`）。

```
285 \scan@allowedtrue }
```

`\macro@name` 现在我们来讨论组合由一个或多个“字母”（可能包括 `@` 符号或其他类别码为 11 的字符）组成的命令名称的宏。

为此，我们将“字母”添加到已有的 `\macro@namepart` 定义中（你可能还记得最初设置为 `\@empty`）。

```
286 \def\macro@name#1{\edef\macro@namepart{\macro@namepart#1}%
```

然后我们抓住下一个单个字符，并让 `\more@macroname` 确定它是属于形成命令名称的字母字符串，还是“非字母”。

```
287 \futurelet\next\more@macroname}
```

`\more@macroname` 如果下一个字符确实是一个“字母”，这将导致另一个 `\macro@name` 的调用，以添加下一个字符。

```
288 \def\more@macroname{\ifcat\noexpand\next a%
```

```
289 \let\next\macro@name
```

否则，它通过调用 `\macro@finish` 来完成索引条目的结尾。

```
290 \else \let\next\macro@finish \fi
```

这里是我们调用了 `\let` 等于 `\next` 的任何宏。

```
291 \next}
```

`\macro@finish` 当我们组装完成形成命令名称的完整“字母”字符串时，我们设置形成整个命令名称的字符，并生成适当的 `\index` 命令（前提是命令名称不在排除列表中）。由于“`\`”已经被排版，因此我们只需输出保存在 `\macro@namepart` 中的所有“字母”。

```
292 \def\macro@finish{%
```

```
293 \macro@namepart
```

然后我们调用 `\ifnot@excluded` 来决定是否需要生成索引条目。使用 `\@tempa` 构造的原因是我们希望在 `\index` 命令中使用 `\macro@namepart` 的扩展。¹⁹

```
294 % \ifnot@excluded
```

```
295 % \edef\@tempa{\noexpand\SpecialIndex{\bslash\macro@namepart}}%
```

```
296 % \@tempa \fi
```

```
297 \maybe@index@macro \macro@namepart
```

```
298 }
```

7.8 索引排除列表²⁰

¹⁹ 命令 `\index` 将在 `\output` 过程中扩展其参数。在这个时候，`\macro@namepart` 可能已经有了新的值。

²⁰ 信息：对于 `\DoNotIndex` 及其调用的宏的不完整注解由 Dave Love 撰写。

以下代码部分采用了新的实现方式，使用了 L^AT_EX3 编程层，其中提供的构造和类型使编程更加轻松。随着时间的推移，我可能会逐步替换掉这份文档中的其他代码部分。

299 \ExplSyntaxOn

`\l__doc_donotindex_seq` 本地序列，保存不应被索引的命令名称（作为字符串）。在 `doc` 元素环境中，该元素被放入序列中，以便在该代码部分中不会被不必要地索引。由于序列是局部的，在环境结束时将恢复该设置，以便在其他地方对该命令进行索引（除非它被全局禁用了索引）。

`\g__doc_idxtype_prop` 全局属性列表，保存所有特殊 `doc` 元素的元素类型。键是不带反斜杠的命令名称，值是 `doc` 元素类型标识符，例如，如果已经设置了长度寄存器的类型，则为 `\texttt{Length}`。`doc` 只索引命令，即以转义字符开头的内容，默认情况下是反斜杠。不以转义字符开头的 `doc` 元素，例如环境，在解析代码时不会被识别，因此不会被自动索引。因此，对于它们来说，在该属性列表中保留它们是没有意义的。

`\doc_dont_index:n` 接受命令（带有反斜杠）的 `clist` 作为输入，并将所有这些命令排除在索引之外。用户可通过 `\DoNotIndex` 使用此命令。

`\ShowIndexingState` 以相当低级的形式显示当前的排除索引列表。

`\RecordIndexType` 此命令接受两个参数：一个命令（带有转义字符）和其类型（即 `\NewDocElement` 声明的第一个必选参数）。如果 `#1` 不应被索引，那么数据将用于记录此命令是该类型。然后使用此信息生成相应的索引条目。显然，早期生成的索引条目将列出错误的类型。因此，此信息还被放置在 `.aux` 文件中，以便在进一步运行的开头使用。

此命令作为任何 `doc` 元素环境的内部执行，因此只需要在某些原因下命令与特殊类型的环境没有相应的情况下显式给出。

`\l__doc_donotindex_seq` 声明。

```
\g__doc_idxtype_prop 300 \seq_new:N \l__doc_donotindex_seq
301 \prop_new:N \g__doc_idxtype_prop
```

`__doc_trace:x` 用于追踪的辅助...

```
302 \cs_new:Npn \__doc_trace:x {
303   \legacy_if:nTF{ doc@debugshow }{ \iow_term:x }{ \use_none:n }
304 }
```

`\doc_dont_index:n` 解析参数为一个命令 `clist`，其中使用 `\MakePrivateLetters`（使得特殊字符被识别为命令名的一部分），并将每个命令的反斜杠删除，转换为字符串后放

`__doc_dont_index_aux:n` 入序列中。`\l__doc_donotindex_seq`.

```
305 \cs_new:Npn \doc_dont_index:n {
```

```

306 \group_begin:
307   \MakePrivateLetters
308   \__doc_dont_index:n
309 }

310 \cs_new:Npn \__doc_dont_index:n #1 {
311   \group_end:

312   \__doc_trace:x{Disable~ indexing~ for~ '\tl_to_str:n{#1}' }

```

通过在 `clist` 中的每个元素上映射函数 `__doc_dont_index_aux:n` 来将命令添加到 `\l__doc_donotindex_seq` 序列中。

```

313 \clist_map_function:nN {#1} \__doc_dont_index_aux:n
314 }

```

我们通过使用命令的名称作为字符串来记录每个命令。这意味着序列中会有更多的标记，但这样可以比较名称而不是“操作”，这很重要，因为不同的命令可能具有相同的含义（例如，它们可能根本没有被定义）。

```

315 \cs_new:Npn \__doc_dont_index_aux:n #1 {
316   \seq_put_right:Nx \l__doc_donotindex_seq {\expandafter\@gobble \string#1}
317 }

```

`\DoNotIndex` 文档级接口

```

318 \cs_set_eq:NN \DoNotIndex \doc_dont_index:n

```

`\ShowIndexingState` 一些可能有用的跟踪信息。

```

319 \def \ShowIndexingState {
320   \__doc_trace:x{Show~ doc~ indexing~ state:}
321   \seq_show:N \l__doc_donotindex_seq
322   \prop_show:N \g__doc_idxtype_prop
323 }

```

`__doc_idxtype_put:Nn` **TODO:** 更改接口命令的名称!

`\RecordIndexType`
`\RecordIndexTypeAux` 这是 `\RecordIndexType` 的内部形式。第一个参数被转换为字符串，其余的处理由 `__doc_idxtype_put:nn` 完成。

```

324 \cs_new:Npn \__doc_idxtype_put:Nn #1#2 {
325   \exp_args:Nx \__doc_idxtype_put:nn { \cs_to_str:N #1 }{#2}

```

我们还在 `.aux` 文件中创建条目，以便此声明在下次运行时立即可用。但是，我们不重用 `__doc_idxtype_put:N`（也称为 `\RecordIndexType`），因为这样会导致每次运行文档时这些行都会翻倍。相反，我们使用 `\RecordIndexTypeAux`，它仅更新数据结构而不写入 `.aux` 文件。

```

326   \protected@write\@auxout{}

```

```

327     {\string\RecordIndexTypeAux {\string#1 }{#2} }
328 }

```

当我们从 .aux 文件执行此代码时，最好不要在 .aux 文件中生成新行。否则，随着时间的推移，这些行会累积。

```

329 \cs_new:Npn \RecordIndexTypeAux #1#2 {
330   \exp_args:Nx \__doc_idxtype_put:nn { \cs_to_str:N #1 }{#2}
331 }

```

类似地，当在运行结束时读取 .aux 文件时，我们应禁用该命令以避免不必要的处理。

```

332 \AtEndDocument{
333   \cs_set_eq:NN \RecordIndexTypeAux \use_none:nn
334 }

```

最后，我们提供用户级接口

```

335 \cs_set_eq:NN \RecordIndexType \__doc_idxtype_put:Nn

```

`__doc_idxtype_put_scan:nn` 当我们想要记录扫描名称的索引类型时，我们不能将其转换为控制序列名称，然后调用 `__doc_idxtype_put:Nn`，因为将其转换为控制序列名称可能会将名称的含义从“未定义”更改为 `\relax`。当处理包含 `\@undefined` 的内核源代码时曾遇到过这个问题：突然间它不再是未定义的了。因此，这里有另一个版本，它仅适用于字符作为输入。由于我们不知道它们是否已经是正确的字符串，所以我们首先确保这种情况成立。

```

336 \cs_new:Npn \__doc_idxtype_put_scan:nn #1#2 {
337   \exp_args:Nf \__doc_idxtype_put:nn { \tl_to_str:n {#1} }{#2}

```

在这种情况下，写入 .aux 文件时也必须附加一个反斜杠。

```

338   \protected@write\@auxout{
339     {\string\RecordIndexTypeAux {\backslash #1 }{#2} }
340 }

```

`__doc_idxtype_put_scan:nn` 这是我们实际需要的，因为字符存储在某些宏中。

```

341 \cs_generate_variant:Nn \__doc_idxtype_put_scan:nn {o}

```

`\record@index@type@save` 这是与代码的其余部分进行交互的接口：

```

342 \cs_set_eq:NN \record@index@type@save \__doc_idxtype_put_scan:on

```

`__doc_idxtype_put:nn` 这个内部命令接受两个参数：作为字符串的命令名称（无反斜杠），以及它的类型（即，`\NewDocElement` 声明的第一个必选参数）。如果 `#1` 不在 `\l__doc_donotindex_seq` 中，它将使用 `#1` 作为键，`#2` 作为其值将这些数据添加到属性列表 `\g__doc_idxtype_prop` 中。如果键已经存在，它的值将被覆

盖。如果以后将该命令标记为不包括在索引中，属性列表设置将保持不变，但只要没有为该命令生成索引，就不会被查询。

注意：该命令假设 #1 已经以字符串形式存在

```
343 \cs_new:Npn \__doc_idxtype_put:nn #1#2 {
```

没有什么神秘的：如果该命令没有被标记为不包括在索引中，则添加属性。额外的 \tl_to_str:n 是一种安全措施，以防输入不是以该形式存在（应该只在输入损坏的情况下才会出现，但是……）

```
344   \exp_args:Nnf
345   \seq_if_in:NnTF \l__doc_donotindex_seq {\tl_to_str:n{#1}}
```

一些跟踪信息 ...

```
346   {
347     \__doc_trace:x{Not~ recording~ index~ type~ for~ '\bslash #1' }
348   }
349   {
350     \__doc_trace:x{Recording~ index~ type~ for~ '\bslash #1' ~ as~ #2 }
```

将数据放入属性列表：

```
351     \prop_gput:Nnn \g__doc_idxtype_prop {#1}{#2}
352   }
353 }
```

\exp_args:co 辅助函数：构造一个函数并用其第一个参数展开一次调用它：

```
354 \cs_new:Npn \exp_args:co #1#2
355   { \cs:w #1 \exp_after:wN \cs_end:\exp_after:wN {#2} }
```

\tl_to_str:o 另一个辅助函数：获取某个记号列表变量，展开它并将其转换为字符串。

```
356 \cs_generate_variant:Nn \tl_to_str:n {o}
```

\maybe@index@macro 这个命令接受一个宏名称（在 macrocode 环境中解析时不带反斜杠）并检查是否应该对其进行索引（即，不在排除列表中），如果应该，则确定索引方式（即，获取其索引类型属性并根据该属性进行正确选择）。

\maybe@index@macro 首先确保参数确实是一个字符串（以确保我们处于定义的情况下），然后将其传递给 __doc_maybe_index_aux:nN 进行处理。第二个参数定义了索引操作：对于多字母宏使用 \SpecialIndex，对于单个字符宏使用 \SpecialShortIndex。

```
357 \cs_new:Npn \__doc_maybe_index:o #1 {
358   \exp_args:Nf \__doc_maybe_index_aux:nN { \tl_to_str:o {#1} }
359                                     \SpecialIndex
360 }
```

这是在旧的非 expl3 代码中调用它的方式：

```
361 \cs_set_eq:NN \maybe@index@macro \__doc_maybe_index:o
```

\maybe@index@short@macro 单个字符宏的处理类似，但这里的索引是由 \SpecialShortIndex 完成的，而
__doc_maybe_index_short:o 且更简单，因为我们知道参数包含一个字符串记号而不是字母。

```
362 \cs_new:Npn \__doc_maybe_index_short:o #1 {  
363   \exp_args:No \__doc_maybe_index_aux:nN #1  
364                                     \SpecialShortIndex  
365 }  
366 \cs_set_eq:NN \maybe@index@short@macro \__doc_maybe_index_short:o
```

__doc_maybe_index_aux:nN 接受一个字符串（表示不带反斜杠的宏），并根据索引进行正确的选择。

```
367 \cs_new:Npn \__doc_maybe_index_aux:nN #1#2 {
```

一些跟踪信息：

```
368   \__doc_trace:x{搜索 '\backslash #1' }
```

如果名称在排除列表中，则不执行任何操作。

```
369   \seq_if_in:NnTF \l__doc_donotindex_seq {#1}  
370   {  
371     \__doc_trace:x{Not~ indexing~ '\backslash #1' }  
372   }
```

否则，检查这个名称是否附加了索引类型属性。

```
373   {  
374     \prop_get:NnNTF \g__doc_idxtype_prop {#1} \l__doc_idxtype_tl
```

如果是，构造并执行 \Code<idxtype>Index²¹，这是在 __doc_maybe_index_aux 中完成的。

```
375   {  
376     \exp_args:Ncno \__doc_maybe_index_aux:Nnn  
377       { Code \tl_use:N \l__doc_idxtype_tl Index }  
378     {code} {\backslash #1}  
379   }
```

否则执行 \SpecialIndex，它是 \CodeMacroIndex{code} 的简写，或执行 \SpecialShortIndex，它处理一些单个字符宏的特殊情况。

```
380   {  
381     \__doc_trace:x{Indexing~ '\backslash #1'\space (\string #2)}  
382     \exp_args:No #2 {\backslash #1}  
383   }  
384 }  
385 }
```

²¹我猜这应该真正是一个内部名称而不是用户级的名称。

\SpecialShortIndex **TODO:** 待记录; 目前混合了旧的和新的, 需要整理

```
386 \cs_new:Npn \SpecialShortIndex #1 {
387   \@SpecialIndexHelper@ #1\@nil
388   \@bsphack
389   \ifdoc@noindex \else
390     \str_case_e:nnF {\@gtempa }
391     {
392       {\cs_to_str:N \^^M } {\def\reserved@a{ \string \space \actualchar }
393                             \def\reserved@b { \space }
394                             \let\reserved@c \@empty }
395     }
```

针对 _ 的修复, 现在我们需要寻找一个真实的空格来处理这个命令序列。

```
395     { ~ } {\def\reserved@a{ \string \space \actualchar }
396             \def\reserved@b { \space }
397             \let\reserved@c \@empty }
398     {\c_left_brace_str} { \def\reserved@a{ \bgroup \actualchar }
399                           \def\reserved@b { \c_left_brace_str }
400                           \def\reserved@c { \noexpand\iffalse
401                                           \c_right_brace_str
402                                           \noexpand\fi }
403     {\c_right_brace_str} { \def\reserved@a{ \egroup \actualchar
404                                           \noexpand\iffalse
405                                           \c_left_brace_str
406                                           \noexpand\fi }
407                           \def\reserved@b { \c_right_brace_str }
408                           \let\reserved@c \@empty }
409     }
```

\verbatimchar 的情况比较棘手。我们不能直接放入普通的 \verb 中, 否则会得到类似 \verb+\++ 这样的结果, 会输出为 “\+”, 而不是 \+. 因此, 我们使用 \verb 只生成反斜杠, 然后在 \verbatimchar 上使用 \texttt。然而, 如果不幸地有人 (比如 Will :-)) 使用了像 & 这样带有 \verbatimchar 特殊类别码的字符, 仅此还不够, 在读取时我们还需使用 \string 处理它。

```
409     {\verbatimchar} { \def\reserved@a{ \quotechar\verbatimchar
410                                           \actualchar }
411                       \let\reserved@b \@empty
412                       \def\reserved@c
413                       { \string\texttt{\string\string\verbatimchar} } }
414     }
415     { \def\reserved@a {\quotechar \@gtempa \actualchar }
416       \def\reserved@b {\quotechar \@gtempa }
417       \let\reserved@c \@empty }
418     \special@index {
```

```

419 \reserved@a
420 \string\verb
421 \quotechar *\verbatimchar \quotechar \bslash
422 \reserved@b
423 \verbatimchar
424 \reserved@c
425 \encapchar code}
426 \fi
427 \@esphack
428 }

```

`__doc_maybe_index_aux:Nnn` 执行作为第一个参数传递的函数，以第二个和第三个参数作为输入。

```

429 \cs_new:Npn \__doc_maybe_index_aux:Nnn #1#2#3 {

```

我们必须小心一点：由于该函数名是动态构造的，可能实际上并不存在（在这种情况下，构造在 $\text{T}_{\text{E}}\text{X}$ 中会生成 `\relax`）。如果是这种情况，我们会引发一个错误；否则，我们会执行该函数（附带一些跟踪信息）：

```

430 \cs_if_exist:NTF #1
431 {
432 \__doc_trace:x{Indexing~ '#3'\space as~
433 \tl_use:N \l__doc_idxtype_tl }
434 #1{#2}{#3}
435 }
436 {
437 \PackageError{doc}{Doc~ element~
438 '\tl_use:N \l__doc_idxtype_tl'~ unknown}%
439
440 {When~ using~ '\string\RecordIndexType'~ the~ type~ must~
441 be~ known~\MessageBreak
442 to~ the~ system,~ i.e.,~ declared~ via~
443 '\string\NewDocElement'\MessageBreak
444 before~ it~ can~ be~ used~ in~ indexing.}
445 }
446 }

```

回到旧式编码风格 ...

```

447 \ExplSyntaxOff

```

7.9 生成索引条目的宏

在这里，我们提供了用于生成索引条目的宏的默认定义；这些宏可以被显式调用，或者由 `\scan@macro` 自动调用。正如已经提到的，这里给出的定义

假定 .idx 文件将由 Chen 的 makeindex 程序处理 — 它们可以重新定义以用于用户喜欢的类似程序。

为了帮助读者在索引中定位条目，所有这些条目都按字母顺序排序，无视起始的 ‘\’；这是通过发出包含 makeindex 的 ‘actual’ 操作符的 \index 命令实现的。后者的默认值是 ‘@’，但在 L^AT_EX 的宏包文件中，这个字符如此流行，因此需要替换另一个字符。这通过一个“索引样式文件”通知给 makeindex；为此函数选择的字符是 =，因此在遇到这个字符时，在 T_EX 命令中也必须特别处理。一个适当的索引样式文件在支持此样式文件的文件中提供，名为 gind.ist，它是通过使用 docstrip 处理从这个源文件中提取模块 gind 生成的。类似的样式文件 gglo.ist 用于对词汇表文件中的变更信息排序，它作为模块 gglo 被提取。首先，我们在 .ist 文件的前面添加了一些信息。

```
448 </package>
449 <+gind | gglo>%% 这是一个 MAKEINDEX 样式文件，
450 <+gind | gglo>%% 应该用于生成文档包中格式化的索引。
451 <+gind>%% 用于文档包的格式化索引。
452 <+gglo>%% 用于文档包的格式化变更历史。
453 <+gind | gglo>%% 下面使用的 TeX 命令在 doc.sty 中定义。
454 <+gind | gglo>%% MAKEINDEX 中类似 ‘level’、‘item_x1’ 的命令在
455 <+gind | gglo>%% Pehong Chen 的《Makeindex, A General Purpose,
456 <+gind | gglo>%% Formatter-Independent Index Processor》中有描述。
457 <+gind | gglo>
```

\actualchar 首先是 \actualchar、\quotechar 和 \levelchar 的定义。注意，我们的默认值并不是在没有样式文件的情况下 makeindex 程序所使用的默认值。

```
\levelchar 458 <*package>
459 \ifundefined{actualchar}{\def\actualchar{=}}{}
460 \ifundefined{quotechar}{\def\quotechar{!}}{}
461 \ifundefined{levelchar}{\def\levelchar{>}}{}
462 </package>
463 <+gind | gglo>actual ’=’
464 <+gind | gglo>quote ’!’
465 <+gind | gglo>level ’>’
466 <*package>
```

\encapchar 对于 \encapchar，makeindex 的默认值没有改变。

```
467 \ifundefined{encapchar}{\def\encapchar{||}}{}
```

\verbatimchar 我们还需要一个特殊字符作为下面定义中 \verb* 命令的分隔符。

```
468 \ifundefined{verbatimchar}{\def\verbatimchar{+}}{}
```


`\@SpecialIndexHelper@` **TODO:** 记录或删除

```
469 \begingroup
470 \catcode'\|=0
471 \catcode'\|=12
472 |gdef|\@SpecialIndexHelper@#1#2|\@nil{%
473   |if |noexpand#1\%
474     |gdef|\@gtempa{#2}%
475   |else
476     |begingroup
477       |escapechar|m@ne
478       |expandafter|gdef|expandafter|\@gtempa|expandafter{|string#1#2}%
479     |endgroup
480   |fi}
481 |endgroup
```

`\SortIndex` 这个宏用于为 `\scan@macro` 遇到的任何单字符命令生成索引条目。第一个参数指定字符的词法顺序，第二个参数给出条目中实际打印的字符。它还可以直接用于生成排序键和实际条目不同的索引条目。

```
482 \def\SortIndex#1#2{%
483   \ifdoc@noindex\else
484     \index{#1\actualchar#2}%
485   \fi
486 }
```

`\LeftBraceIndex` 这两个宏修复了与 `makeindex` 相关的问题。请注意使用 `\iffalse\fi` 这种‘技巧’来同时满足 `TEX` 和 `makeindex` 的需求。当写入到 `.idx` 文件时，`TEX` 将同时看到两个大括号（因此我们得到了平衡的文本）。`makeindex` 也会看到平衡的大括号，但当实际的索引条目再次被 `TEX` 处理时，`\iffalse\fi` 之间的大括号将消失。

```
487 \@ifundefined{LeftBraceIndex}{\def\LeftBraceIndex{%
488   \special@index{\bgroup\actualchar
489     \string\verb% % 为了糊弄 emacs 的代码高亮
490     \quotechar*\verbatimchar
491     \quotechar\bslash{\verbatimchar\string\iffalse}\string\fi}}{}
492
493 \@ifundefined{RightBraceIndex}{\def\RightBraceIndex{%
494   \special@index{\egroup\actualchar\string\iffalse{\string\fi
495     \string\verb% % 为了糊弄 emacs 的代码高亮
496     \quotechar*\verbatimchar\quotechar\bslash{\verbatimchar}}{}}
```

`\PercentIndex` 默认情况下，我们假设正在使用修复了百分号 bug 的版本的 `makeindex`。

```
497 \ifundefined{PercentIndex}
498   {\def\PercentIndex{\it@is@a\percentchar}}{}
```

`\OldMakeindex` 这是解决 `makeindex` 中百分号 bug 的一个方案。宏 `\percentchar` 表示一个 `\percentchar` %₁₂。在包或驱动文件中调用此宏会适当地设置事务。

```
499 \def\OldMakeindex{\def\PercentIndex{%
500   \special@index{\quotechar\percentchar\actualchar
501     \string\verb% % 为了糊弄 emacs 的代码高亮
502     \quotechar*\verbatimchar\quotechar\bslash
503     \percentchar\percentchar\verbatimchar}}}
504 {\catcode'\%=12 \gdef\percentchar{}}}
```

`\it@is@a` 这个宏应该为给定的字符产生一个正确的 `\SortIndex` 条目。由于这个字符可能被索引程序识别为‘命令’字符，所有字符都用 `\quotechar` 引用起来。

```
505 \def\it@is@a#1{\special@index{\quotechar #1\actualchar
506   \string\verb% % 为了糊弄 emacs 的代码高亮
507   \quotechar*\verbatimchar
508   \quotechar\bslash\quotechar#1\verbatimchar}}
```

7.10 重新定义 index 环境

`\IndexMin` (*dimen*) 如果正在使用 `multicol`，在索引开始时我们计算当前页面的剩余空间；如果 `IndexColumns` (*counter*) 大于 `\IndexMin`，则索引的第一部分将放置在可用空间中。设置的列数由计数器 `\c@IndexColumns` 控制，可以使用 `\setcounter` 声明更改它。

```
509 \newdimen\IndexMin      \IndexMin      = 80pt
510 \newcount\c@IndexColumns \c@IndexColumns = 3
```

`theindex` (*env.*) 现在，如果适用，我们开始多列机制。我们使用上面声明的 `LATEX` 计数器 `\c@IndexColumns` 表示列数，并插入‘索引序言’文本（可能包含 `\section` 调用等）。有关示例，请参阅默认定义。

```
511 \ifdoc@multicol
512   \RequirePackage{multicol}
513   \renewenvironment{theindex}
514     {\begin{multicols}\c@IndexColumns[\index@prologue][\IndexMin]}%
```

然后我们进行一些最后的分配，以读取各个索引 `\item`，并最后忽略任何初始空格。

```
515   \IndexParms \let\item\@idxitem \ignorespaces}%
```

`\endtheindex` 在索引的结尾，我们只需要结束 `multicols` 环境。

```
516     {\end{multicols}}
```

如果我们不能使用 `multicols`，我们会警告用户并使用基本上来自 `article.sty` 的环境。

```
517 \else
```

```
518   \def\theindex{\@restonecoltrue\if@twocolumn\@restonecolfalse\fi
```

```
519   \columnseprule \z@ \columnsep 35\p@
```

```
520   \twocolumn[\index@prologue]%
```

```
521   \IndexParms \let\item\@idxitem \ignorespaces}
```

```
522   \def\endtheindex{\if@restonecol\onecolumn\else\clearpage\fi}
```

```
523 \fi
```

这里是必要的 `makeindex` 声明。我们使用 `\scan@allowedfalse\n` 禁用索引中宏名称的扫描，以避免递归。

```
524 </package>
```

```
525 <+gind>preamble
```

```
526 <+gind>"\n \\\begin{theindex} \n \\\makeatletter\scan@allowedfalse\n"
```

```
527 <+gind>postamble
```

```
528 <+gind>"\n\n \\\end{theindex}\n"
```

```
529 <*package>
```

`\IndexPrologue` `\IndexPrologue` 宏用于在索引上方的文档中放置一个简短的消息。它通过重定义 `\index@prologue` 实现，默认文本存储在这个宏中。最好将其定义为 `\long` 宏，以允许在其参数中使用 `\par` 命令。

```
530 \long\def\IndexPrologue#1{\@bsphack\def\index@prologue{#1}\@esphack}
```

现在我们测试默认值是否已经被其他包文件定义。如果没有，我们就定义它。

```
531 \@ifundefined{index@prologue}
```

```
532   {\def\index@prologue{\section*{Index}%
```

```
533     \markboth{Index}{Index}%
```

```
534     Numbers written in italic refer to the page
```

```
535     where the corresponding entry is described;
```

```
536     numbers underlined refer to the
```

```
537     \ifcodeline@index
```

```
538     code line of the
```

```
539     \fi
```

```
540     definition; numbers in roman refer to the
```

```
541     \ifcodeline@index
```

```
542     code lines
```

```
543     \else
```

```
544     pages
```

```

545             \fi
546             where the entry is used.
547         }}{}

```

`\IndexParms` 这些是用于格式化索引条目的最后一刻分配。它们在单独的宏中定义，这样用户可以替换不同的定义。我们首先定义控制行间距和两列之间间隔的各种参数。整个索引设置为 `\small` 大小。

```

548 \@ifundefined{IndexParms}
549     {\def\IndexParms{%
550         \parindent \z@
551         \columnsep 15pt
552         \parskip 0pt plus 1pt
553         \rightskip 15pt
554         \mathsurround \z@
555         \parfillskip=-15pt
556         \small

```

`\@idxitem` 索引条目采用悬挂缩进，对于可能超过一行的任何条目。

```

\subitem 557         \def\@idxitem{\par\hangindent 30pt}%

```

`\subsubitem` 索引中的任何子条目都以其主标题下的 15pt 缩进格式化。

```

558         \def\subitem{\@idxitem\hspace*{15pt}}%

```

而子子条目则进一步缩进 10pt。

```

559         \def\subsubitem{\@idxitem\hspace*{25pt}}%

```

`\indexspace` 在每个新的字母顺序开始之前，`makeindex` 程序会生成 `\indexspace`。在这最后一个定义之后，我们结束了 `\@ifundefined` 和 `\IndexParms` 的定义。

```

560         \def\indexspace{\par\vspace{10pt plus 2pt minus 3pt}}%
561     }}{}

```

`\efill` 这个对 `\efill` 的定义是用于索引条目之后没有跟随文本的情况（例如，“*see*”条目）。它只是确保当前行被填充，避免“`Underfull \hbox`”的警告信息。

```

562 \def\efill{\hfill\nopagebreak}%
563 </package>
564 <+gind|gglo>item_x1    "\\efill \n \\subitem "
565 <+gglo>item_x2        "\\ "
566 <+gind>item_x2        "\\efill \n \\subsubitem "
567 <*package>

```

`\pfill` 下面的定义提供了 `\pfill` 命令；如果在索引样式文件中将其指定给 `makeindex`，作为出现在索引条目后的分隔符，那么引用页码之前的间隙将用点号填充，在点号两端插入了一点白色空间。如果行被打断，点号将出现在两行上。

```

568 \def\pfill{\unskip~%
569           \leaders\hbox to.6em{\hss.\hss}\hfill
570           \penalty500\strut\nobreak
571           \leaders\hbox to.6em{\hss.\hss}\hfil
572           ~\ignorespaces}%
573 </package>
574 <+gind|gglo>delim_0  "\\pfill "
575 <+gind|gglo>delim_1  "\\pfill "
576 <+gind|gglo>delim_2  "\\pfill "
577 <*package>

```

`*` 这里是 `*` 宏的定义。它在这组宏中没有使用。

```

578 \def\*{\leavevmode\lower.8ex\hbox{$\,\widetilde{\ }},\$}}

```

`\main` 宏名称的定义条目在 `\index` 命令中被标记为字符串 `|main`²²；`makeindex` 处理这个字符串，使得在包含宏的定义所在页码上调用 `\main` 宏会添加下划线。

```

579 \@ifundefined{main}{\def\main#1{\underline{#1}}}{\}

```

`\usage` `\usage` 宏用于指示描述宏用法的条目。相应的页码将以斜体设置。

```

580 \@ifundefined{usage}{\def\usage#1{\textit{#1}}}{\}

```

`\code` `\code` 宏用于指示不是主要条目的代码行索引条目。默认情况下，我们不对它们做任何特殊处理。

```

581 \@ifundefined{code}{\def\code#1{#1}}{\}

```

`\PrintIndex` 这与 `makeidx` 包中的 `\printindex` 相同。

```

582 \def\PrintIndex{\@input@{\jobname.ind}%
583           \global\let\PrintIndex\@empty}

```

我们希望索引（以及更改列表）的标题根据下一个条目块的首字符而变化，并且必须相应地指示给 `makeindex`。不幸的是，在 `makeindex` 的版本 2.4 和 2.11 之间的规范发生了变化。我们提供了两种做法，但不幸的是，这将始终从 `makeindex` 产生警告信息。这是对于旧版本的：

²²使用当前定义的 `\encapchar` 替换了 `|`

```

584 </package>
585 <+gind,gglo>% 下面几行在运行 Makeindex 时会产生一些警告,
586 <+gind,gglo>% 因为它们尝试覆盖两个不同版本的程序:
587 <+gind,gglo>lethead_prefix    "{\\bfseries\\hfil "
588 <+gind,gglo>lethead_suffix    "\\hfil}\\nopagebreak\\n"
589 <+gind>lethead_flag           1
590 <+gglo>lethead_flag           0

```

对于更新的版本, 则使用以下定义:

```

591 <+gind,gglo>heading_prefix    "{\\bfseries\\hfil "
592 <+gind,gglo>heading_suffix    "\\hfil}\\nopagebreak\\n"
593 <+gind>headings_flag          1
594 <+gglo>headings_flag          0
595 <*package>

```

7.11 处理变更历史²³

为了提供变更历史记录, 引入了`\changes` 命令。这个命令接受三个参数, 分别是文件的版本号、变更日期以及变更的细节。第二个参数通常被忽略, 但其他两个会被写入并可用于生成变更历史, 最终打印在文档末尾。但需要注意的是, 陈先生标准的 `makeindex` 程序的旧版本将任何文本字段限制在 64 个字符; 因此, 重要的是第二个和第三个参数的字符数总共不应超过 61 个 (为了在日期周围放置括号)。

`\changes` `\changes` 命令的输出进入 *<Glossary_File>*, 因此使用正常的 `\glossaryentry` 命令。²⁴ 因此, `makeindex` 或类似的程序可以用来处理输出, 生成一个排序的“词汇表”。`\changes` 命令开始时采取通常的措施隐藏其间距, 然后重新定义 `\protect` 以在生成的 `\indexentry` 命令的参数内使用。

我们几乎将 `\sanitize` 中找到的所有字符重新编码为字母, 因为使用一些使某些字符变为活动字符的特殊包可能会在将其条目写入文件时影响 `\changes` 命令。然而, 我们必须将 `%` 保留为注释, `␣` 作为 *<space>*, 否则会出现混乱。当然, `\` 应该作为转义字符可用。

```

596 \def\changes{\@bsphack\beginngroup\@sanitize
597   \catcode'\z@ \catcode'\ 10 \MakePercentIgnore
598   \changes@}
599 \def\changes@#1#2#3{%
600   \protected@edef\@tempa{\noexpand\glossary{#1%

```

²³整个章节由 Brian HAMILTON KELLY 提出。他还记录和调试了宏以及这个包的许多其他部分。

²⁴需要注意的是, 最近在 L^AT_EX 2.09 中更改了 `.glo` 文件中的命令名称, 从 `\indexentry` 变为 `\glossaryentry`。因此, 需要一个名为 `gglo.ist` 的特殊 `makeindex` 样式文件来正确处理这个文件。

如果需要，我们也展示变更日志中的日期（在版本号之后）。

```
601          \ifdoc@reportchangedates
602          \space -- #2\fi
603          \levelchar
```

如果宏 `\saved@macroname` 不包含任何宏名称（即为空），当前的变更条目是在顶层完成的。在这种情况下，我们在其前面加上 `\generalname`。

```
604          \ifx\saved@macroname\@empty
```

在条目开头放一个！可能会在排序过程中将此条目移到最前面。

```
605          \quotechar!%
606          \actualchar
607          \generalname
608          \else
609          \saved@indexname
610          \actualchar
611          \string\verb% % 为了糊弄 emacs 的代码高亮
612          \quotechar*%
613          \verbatimchar\saved@macroname
614          \verbatimchar
615          \fi
616          : \levelchar #3}%
617 \@tempa\endgroup\@esphack}
```

`\saved@macroname` 条目按照最近引入的宏名称的名称（即在最近的 `\begin{macro}` 命令中的名称）进行排序以方便查找。因此，我们提供 `\saved@macroname` 来记录该参数，并在 `\changes` 在 `macro` 环境外使用时提供默认定义。（这是一个狡猾的 hack，以便将这些条目放在排序列表的开头！它能够正常工作，前提是没有宏名称以！或 " 开头。）

```
618 \def\saved@macroname{}
```

`\saved@indexname` 用于索引的宏名称（或者原本就没有反斜杠的环境名称）。

```
619 \def\saved@indexname{}
```

`\generalname` 这个宏保存在顶层变更条目之前放置的字符串。

```
620 \def\generalname{General}
```

`\RecordChanges` 为了将变更写入（到一个 `.glo`）文件中，我们定义 `\RecordChanges` 来调用 L^AT_EX 的通常 `\makeglossary` 命令。

```
621 \let\RecordChanges\makeglossary
```

`\GlossaryMin` (*dimen*) 剩余的宏都是用于 `theindex` 环境的类比。当开始创建词汇表时，我们计算当前页面底部剩余的空间；如果这个空间大于 `\GlossaryMin`，那么词汇表的第一部分将放置在可用空间中。设置的列数由计数器 `\c@GlossaryColumns` 控制，可以使用 `\setcounter` 声明进行更改。

```
622 \newdimen\GlossaryMin      \GlossaryMin      = 80pt
623 \newcount\c@GlossaryColumns \c@GlossaryColumns = 2
```

`theglossary` (*env.*) 环境 `theglossary` 的定义方式与 `theindex` 环境相同。

```
624 \ifdoc@multicol
625   \newenvironment{theglossary}{%
626     \begin{multicols}\c@GlossaryColumns
627                           [\glossary@prologue][\GlossaryMin]%
628     \GlossaryParms \let\item\@idxitem \ignorespaces}%
629   {\end{multicols}}
630 \else
631   \newenvironment{theglossary}{%
632     \@restonecoltrue\if@twocolumn\@restonecolfalse\fi
633     \columnseprule \z@ \columnsep 35\p@
634     \twocolumn[\glossary@prologue]%
635     \GlossaryParms \let\item\@idxitem \ignorespaces}
636   {\if@restonecol\onecolumn\else\clearpage\fi}
637 \fi
```

这是与索引相同的 `makeindex` 声明，禁用了扫描功能。

```
638 \</package>
639 \<+glo>preamble
640 \<+glo>"\n \<\begin{theglossary}> \n
641 \<+glo>    \<\makeatletter>\<\scan@allowedfalse>\n"
642 \<+glo>postamble
643 \<+glo>"\n\n \<\end{theglossary}>\n"
```

如果您使用的是最新的 L^AT_EX，这与 `gind.ist` 的不同是必要的。

```
644 \<+glo>keyword "\<\glossaryentry>
645 \<*package>
```

`\GlossaryPrologue` `\GlossaryPrologue` 宏用于在文档中的词汇表上方放置一条简短的消息。它通过重新定义 `\glossary@prologue` 实现，这个宏保存默认文本。我们最好将其定义为长宏，以允许其参数中包含 `\par` 命令。

```
646 \long\def\GlossaryPrologue#1{\@bsphack
647                               \def\glossary@prologue{#1}%
648                               \@esphack}
```


现在我们测试默认值是否已被其他包文件定义。如果没有，我们就定义它。

```
649 \ifundefined{glossary@prologue}
650     {\def\glossary@prologue{\section*{{Change History}}%
651         \markboth{{Change History}}{{Change History}}%
652         }}{}}
```

`\GlossaryParms` 除非用户另有指定，否则我们将使用与索引相同的参数设置变更历史，只是我们将其设置为类似左对齐的方式，因为它包含的文本通常在小列中不太容易断行。

```
653 \ifundefined{GlossaryParms}{\let\GlossaryParms\IndexParms
654     \expandafter\def\expandafter\GlossaryParms\expandafter{\GlossaryParms
655         \rightskip 15pt plus 1fil
656         \parfillskip -15pt plus -1fil\relax}
657 }}
```

`\PrintChanges` 要读取并打印排序后的变更历史，只需将 `\PrintChanges` 命令放在您的包文件中的最后一个（注释掉的，因此在文件通过文档过程时执行）命令。或者，该命令可以作为 `\MaybeStop` 命令的一个参数，尽管如果只打印描述，可能不需要变更历史。

该命令假设 `makeindex` 或其他某个程序已经处理了 `.glo` 文件以生成排序后的 `.gls` 文件。

```
658 \def\PrintChanges{\@input@{\jobname.gls}%
659     \global\let\PrintChanges\@empty}
```

7.12 花哨的功能

`\MaybeStop` 如果 `\AlsoImplementation` 生效，整个文档，包括代码部分，将被排版。这 `\Finale` 是默认设置。

`\AlsoImplementation` 660 `\newcommand\AlsoImplementation{%`

`\OnlyDescription` 为了实现这一点，我们必须以 `\Finale` 处排版其参数。为此，我们将其参数保存在宏 `\Finale` 中。

```
661     \long\def\MaybeStop##1{\@bsphack\gdef\Finale{##1%
```

但是 `\Finale` 将在文件的最后调用。这正是我们想要知道文件是否损坏的时刻。因此我们也在此时调用 `\check@checksum`。

```
662         \check@checksum}%
```

另一方面：`\MaybeStop` 或多或少是描述和代码之间的分界点。所以我们开始查找文档文件的校验和，调用 `\init@checksum`。

```

663          \init@checksum
664          \@esphack}%
665      }

```

由于 `\AlsoImplementation` 应该是默认设置，我们执行它，从而给 `\MaybeStop` 赋予所需的含义。

```

666 \AlsoImplementation

```

当用户在驱动文件中放置 `\OnlyDescription` 声明时，文档应仅排版到 `\MaybeStop` 处。因此，我们必须重新定义此宏。

```

667 \def\OnlyDescription{\@bsphack\long\def\MaybeStop##1{%

```

在这种情况下，`\MaybeStop` 的参数应该被设置，之后 `TEX` 应该停止读取此文件。因此，我们用以下方式结束此宏：

```

668          ##1\endinput}\@esphack}

```

如果没有给出 `\MaybeStop` 命令，我们会静默地忽略 `\Finale` 的调用。

```

669 \let\Finale\relax

```

`\StopEventually` `\StopEventually` 是 `\MaybeStop` 的旧错误名称。我们需要使用 `\def`（即扩展），因为 `\MaybeStop` 偶尔会被重新定义。

```

670 \def\StopEventually{\MaybeStop}

```

`\meta` `\meta` 宏有点棘手。我们希望允许在参数的空格处断行，但我们不希望在空格之间断行。过去，这是通过以参数被扫描时使 `␣` 成为活动字符来实现的。然后单词被扫描到 `\hbox` 中。活动的 `␣` 将结束前一个 `\hbox`，添加一个普通空格，并打开一个新的 `\hbox`。这样，断行只能发生在空格处。这种方法的缺点是 `\meta` 既不健壮，也不能被 `\protect`。新的实现解决了这个问题，它以根本不同的方式定义了 `\meta`：我们通过定义一个没有与之关联模式的 `\language` 来阻止连字化，并在尖括号中使用这个语言来排版单词。

```

671 \ifx\l@nohyphenation\undefined
672   \newlanguage\l@nohyphenation
673 \fi
674 \DeclareRobustCommand\meta[1]{%

```

由于 `\meta` 的旧实现可以在数学中使用，我们最好确保新实现也可以。因此，我们在 `\langle` 和 `\rangle` 周围使用 `\ensuremath`。但这还不够：如果 `\meta@font@select` 展开为 `\itshape`，在数学模式下使用它将会失败。因此，在这种情况下，我们将整个内容隐藏在 `\nfss@text` 盒子中。

```

675   \ensuremath\langle
676   \ifmmode \expandafter \nfss@text \fi

```

```

677      {%
678      \meta@font@select
679      \edef\meta@hyphen@restore
680      {\hyphenchar\the\font\the\hyphenchar\font}%
681      \hyphenchar\font\m@ne
682      \language\l@nohyphenation
683      #1\/%
684      \meta@hyphen@restore
685      }\ensuremath\rangle
686 }

```

`\meta@font@select` 使 `\meta` 内部使用的字体可定制。

```

687 \def\meta@font@select{\itshape}

```

`\IndexInput` 接下来的这个宏可用于读取一个单独的文件（可能是一个没有通过此方式文档化的包文件），并以纯文本方式设置它，同时扫描宏名称并对后者进行索引。这可以是准备为所读取的文件生成文档的有用第一步。

```

688 \def\IndexInput#1{%

```

我们开始设置一个组,并初始化一个 `\trivlist`,就像一个 `\begin{macrocode}` 命令通常所做的那样。

```

689      \begingroup \macrocode

```

我们还使间距的行为与 `macrocode` 环境相同，因为否则所有空格都将被显式显示。

```

690      \frenchspacing \@vobeyspaces

```

接着只需要读取指定的文件，并完成 `\trivlist`。

```

691      \input{#1}\endmacrocode

```

当然，我们也需要结束这个组。

```

692      \endgroup}

```

`\maketitle` 生成标题的宏很容易进行修改，以便可以多次使用（例如一篇具有多个标题的文章）。在原始版本中，各种宏在使用后被隐藏起来，使用 `\relax`。我们必须取消任何可能放入 `\@thanks` 等中的内容，否则所有标题都会保留任何之前的这种设置！

```

693 \def\maketitle{\par
694      \begingroup \def \thefootnote {\fnsymbol {footnote}}%

```

```

695 \setcounter {footnote}\z@
696 \def\@makefnmark{\hbox to\z@{\$m@th^{\@thefnmark}$\hss}}%
697 \long\def\@makefnmark##1{\parindent 1em\noindent
698 \hbox to1.8em{\hss\$m@th^{\@thefnmark}$}\##1}%
699 \if@twocolumn \twocolumn [\@maketitle ]%
700 \else \newpage \global \@topnum \z@ \@maketitle \fi

```

对于特殊的格式要求（比如在 TUGboat 中），我们使用 `titlepage` 页面样式；稍后会将其定义为 `plain`，除非已经有定义，例如由 `ltugboat.sty`。

```

701 \thispagestyle{titlepage}\@thanks \endgroup

```

如果驱动文件文档化了许多文件，我们不希望一个标题的部分传播到下一个，因此我们必须取消这些内容：

```

702 \setcounter {footnote}\z@
703 \gdef\@date{\today}\gdef\@thanks{}%
704 \gdef\@author{}\gdef\@title{}

```

`\ps@titlepage` 当多篇文章被串联成期刊时，例如，这些文档的标题页通常不会被格式化得不同。因此，类似 `ltugboat` 这样的类可以提前定义此宏。然而，如果不存在这样的定义，我们就使用页面样式 `plain` 作为标题页。

```

705 \@ifundefined{ps@titlepage}
706 {\let\ps@titlepage=\ps@plain}{}

```

`\MakeShortVerb` 这安排了对 `\verb` 的缩写，如果你接着使用 `\MakeShortVerb{\<c>}`，之后使用 `<c><text><c>` 就相当于 `\verb<c><text><c>`。²⁵ 此外，将 `<c>` 设置为活动字符，以供 `verbatim` 和 `macrocode` 环境使用。特别注意以下定义是全局的。首先我们做的事情（不一定是第一件事）是用 `\add@special` 在 `\dospecials` 和 `\@sanitize` 中记录——假定是新的特殊字符。

一些不谨慎的用户可能会第二次使用 `\MakeShortVerb`，我们最好防范这种情况。我们假设如果绑定了一个控制序列 `\cc<c>`，那么这种情况就发生了，这个名称被另一个模块使用的可能性很低。我们将在下面输出一个警告，这样如果程序员读取了 LOG 文件，可能会注意到潜在的错误。（应该使用模块内部名称，不过。）

`\MakeShortVerb*` 这安排了对 `\verb*` 的缩写，如果你接着使用 `\MakeShortVerb*{\<c>}`，之后使用 `<c><text><c>` 就相当于 `\verb*<c><text><c>`。

```

707 </package>
708 <*package | shortvrb>
709 \def\MakeShortVerb{%

```

²⁵警告：本节余下的评论是由 Dave Love 编写的。

```

710 \ifstar
711   {\def\@shortvrbbdef{\verb*}\@MakeShortVerb}%
712   {\def\@shortvrbbdef{\verb}\@MakeShortVerb}}

```

`\@MakeShortVerb`

```

713 \def\@MakeShortVerb#1{%
714   \expandafter\ifx\csname cc\string#1\endcsname\relax

715   \@shortvrbbinfo{Made }{#1}\@shortvrbbdef
716   \add@special{#1}%

```

然后字符的当前类别码被存储在 `\cc\⟨c⟩` 中。

```

717   \expandafter
718   \xdef\csname cc\string#1\endcsname{\the\catcode'#1}%

```

该字符被拼接到定义中，使用与 `\verb`（例如）中相同的技巧，在一个组中激活 `~`。

```

719   \begingroup
720     \catcode'\~\active \lccode'\~'#1%
721     \lowercase{%

```

将字符的旧含义记录在 `\ac\⟨c⟩` 中，然后为其赋予新含义。

```

722     \global\expandafter\let
723       \csname ac\string#1\endcsname~%
724     \expandafter\gdef\expandafter~\expandafter{\@shortvrbbdef~}%
725   \endgroup

```

最后，字符被激活。

```

726   \global\catcode'#1\active

```

如果我们怀疑 `⟨c⟩` 已经是一个短引用，我们通知用户。现在他或她需要对任何可能出错的地方负责……

```

727   \else

728   \@shortvrbbinfo\@empty{#1 already}%
729       {\@empty\verb% % 为了糊弄 emacs 的代码高亮
730        (*)}%
731   \fi}

```

`\DeleteShortVerb` 这里是取消 `\MakeShortVerb` 的方法，例如在需要在抄录环境之外使用字符的区域。它安排适当地改变 `\dospecials` 和 `\@sanitize`，恢复保存的类别码，并且如果必要的话，恢复字符的含义（由 `\MakeShortVerb` 存储）。如果类别码没有被 `\cc\⟨c⟩`（由 `\MakeShortVerb`）存储，则命令会被默默忽略。

```

732 \def\DeleteShortVerb#1{%
733   \expandafter\ifx\csname cc\string#1\endcsname\relax

```

```

734 \shortvrbinf{\empty{#1 not}%
735             {\empty\verb% % 为了糊弄 emacs 的代码高亮
736             (*}}%
737 \else
738 \shortvrbinf{Deleted }{#1 as}%
739             {\empty\verb% % 为了糊弄 emacs
740             % 代码高亮
741             (*}}%
742 \rem@special{#1}%
743 \global\catcode'#1\csname cc\string#1\endcsname

```

我们不能忘记重置 `\cc\langle c \rangle`，否则在 `\MakeShortVerb` 中检查重复定义时将无法工作。

```

744 \global \expandafter\let \csname cc\string#1\endcsname \relax
745 \ifnum\catcode'#1=\active
746 \begingroup
747 \catcode'\~\active \lccode'\~'#1%
748 \lowercase{%
749 \global\expandafter\let\expandafter~%
750 \csname ac\string#1\endcsname}%
751 \endgroup \fi \fi}

```

`\shortvrbinf` 消息通知的辅助函数。

```

752 \def\shortvrbinf#1#2#3{%
753 \shortverb \PackageInfo{shortverb}{%
754 \!shortverb \PackageInfo{doc}{%
755 #1\expandafter\@gobble\string#2 a short reference
756 for \expandafter\string#3}}

```

`\add@special` 这个辅助宏将其参数添加到 `\dospecials` 宏中，该宏通常被抄录宏使用，用于改变当前活动字符的类别码。我们需要将 `\do\langle c \rangle` 添加到 `\dospecials` 的展开中，如果已经存在该字符，则移除，以避免多次复制，如果 `\MakeShortVerb` 没有被 `\DeleteShortVerb` 平衡（如果使用 `\dospecials` 的任何内容关心重复）。

```

757 \def\add@special#1{%
758 \rem@special{#1}%
759 \expandafter\gdef\expandafter\dospecials\expandafter
760 {\dospecials \do #1}%

```

类似地，我们必须将 `\@makeother\langle c \rangle` 添加到 `\@sanitize` 中（在诸如 `\index` 等地方使用，以重新设定除了大括号外的所有特殊字符的类别码）。

```

761 \expandafter\gdef\expandafter\@sanitize\expandafter

```

```
762     {\@sanitize \@makeother #1}}
```

`\rem@special` `\add@special` 的反函数稍微棘手一些。如果 `\do` 的参数是感兴趣的字符，则重新定义为展开为无内容，否则简单地展开为参数。然后我们可以重新定义 `\dospecials` 为其自身的展开。`=‘##1` 后面的空格防止扩展为 `\relax` !

```
763 \def\rem@special#1{%
764   \def\do##1{%
765     \ifnum‘#1=‘##1 \else \noexpand\do\noexpand##1\fi}%
766   \xdef\dospecials{\dospecials}%
```

Fixing `\@sanitize` 是相似的，只是我们需要重新定义 `\@makeother`，显然这需要在在一个组内完成。

```
767   \begingroup
768     \def\@makeother##1{%
769       \ifnum‘#1=‘##1 \else \noexpand\@makeother\noexpand##1\fi}%
770     \xdef\@sanitize{\@sanitize}%
771   \endgroup}
772 \</package | shortvrb>
773 \<*package>
```

7.13 提供校验和字符表²⁶

`\init@checksum` 校验和机制通过计算宏代码中反斜杠的数量来工作。这个初始化计数的函数（在 `\MaybeStop` 中调用）。

```
774 \def\init@checksum{\relax
775   \global\slash@cnt\z@}
```

`\check@checksum` 此函数报告校验和与文件中声明的值（`\slash@cnt`）进行比较。它从 `\Finale` 中调用（如果没有被重新定义的话）。

```
776 \def\check@checksum{\relax
777   \ifnum\check@sum>\m@ne
778     \ifnum\check@sum=\z@
779       \typeout{*****}%
780       \typeout{* This macro file has no checksum!}%
781       \typeout{* The checksum should be \the\slash@cnt!}%
782       \typeout{*****}%
783     \else
784       \ifnum\check@sum=\slash@cnt
```

²⁶提示：本节中的评论由 Dave Love 撰写。

```

785         \typeout{*****}%
786         \typeout{* Checksum passed *}%
787         \typeout{*****}%
788     \else
789         \PackageError{doc}{Checksum not passed
790             (\the\check@sum<>\the\bslash@cnt)}%
791         {The file currently documented seems to be wrong.^^J%
792         Try to get a correct version.}%
793     \fi
794 \fi
795 \fi
796 \global\check@sum\m@ne}

```

`\check@sum` (*counter*) 我们需要定义计数器, `\bslash@cnt` 用于计算反斜杠的数量, `\check@sum` 用 `\bslash@cnt` (*counter*) 于文件中声明的值 (如果有的话)。负值意味着没有校验和检查, 这是默认值。

```

797 \newcount\check@sum          \check@sum = \m@ne
798 \newcount\bslash@cnt        \bslash@cnt = \z@

```

`\Checksum` 这是设置 `\check@sum` 的接口。

```

799 \def\Checksum#1{\@bspack\global\check@sum#1\relax\@espack}

```

`\step@checksum` 当在宏代码中遇到反斜杠时, 此函数会增加计数。

```

800 \def\step@checksum{\global\advance\bslash@cnt\@ne}

```

`\CharacterTable` 用户接口到字符表检查进行了一些 `\catcode` 设置, 然后将下面的表与存储的版本进行比较。我们需要在表中将 `@` 设置为 “other” 类别码, 因为在正常文档中读取时, 它通常以这种方式返回。为了仍然有一个私有的字母, 我们使用 `~`。作为 “字母”, `~` 不会对后面的空格造成影响, 因为它出现在表的最后, 所以不会吞掉后面的空格。

```

801 \def\CharacterTable{\begingroup \CharTableChanges \character@table}

```

`\character@table` 这段代码用于比较表格并报告结果。请注意, 以下代码被包含在一个以 `~` 设为字母类别码的分组内。

```

802 \begingroup
803   \catcode'\~ = 11
804   \gdef\character@table#1{\def\used~table{#1}%
805       \ifx\used~table\default~table
806           \typeout{*****}%
807           \typeout{* Character table correct *}%
808           \typeout{*****}%

```



```

809      \else
810          \PackageError{doc}{Character table corrupted}
811              {\the\wrong@table}
812      \show\default~table
813      \show\used~table
814  \fi
815  \endgroup}

```

`\CharTableChanges` 当读取字符表时，我们需要使用固定的 `\catcode` 集合来扫描它。下面的参考表是根据 T_EX 的正常 `\catcode` 来定义的，即 `@` 是“other”类型，并且唯一的“letter”类型的标记是常规字母表中的字母。如果出于某种原因，其他字符被设置为“字母”，那么在检查表之前，它们的 `\catcode` 需要恢复。否则，表中的空格会被吞掉，即使它们实际上是相等的，我们也会得到表不同的信息。因此，`\CharTableChanges` 被设置为在本地将这些“字母”的 `\catcode` 恢复为“other”。

```

816  \global\let\CharTableChanges\@empty

```

`\default~table` 这是表格应该（除了空格之外）的样子。

```

817  \makeatother
818  \gdef\default~table
819      {Upper-case   \A\B\C\D\E\F\G\H\I\J\K\L\M\N\O\P\Q\R\S\T\U\V\W\X\Y\Z
820      Lower-case   \a\b\c\d\e\f\g|h|i\j\k\l|m\n\o\p\q\r\s\t\u\v\w\x\y\z
821      Digits       \0\1\2\3\4\5\6\7\8\9
822      Exclamation  \!      Double quote  \"      Hash (number) \#
823      Dollar       \$      Percent      \%      Ampersand    &
824      Acute accent \'      Left paren   \ (      Right paren  \ )
825      Asterisk     *      Plus         +      Comma        ,
826      Minus        -      Point        .      Solidus      /
827      Colon        :      Semicolon   ;      Less than    <
828      Equals       =      Greater than >      Question mark \?
829      Commercial at \@    Left bracket \[      Backslash    \
830      Right bracket \]    Circumflex  ^      Underscore   _
831      Grave accent  `      Left brace  {      Vertical bar |
832      Right brace  }      Tilde       ~
833  \endgroup

```

`\wrong@table` 我们需要在出现问题时提供帮助信息。

```

834  \newhelp\wrong@table{Some of the ASCII characters are corrupted.^~J
835      I now \string\show\space you both tables for comparison.}

```

7.14 将行号附加到代码行²⁷

本节中的代码允许索引条目引用代码行号——在 `macro` 环境中宏代码的第一行的编号。

`\codeline@index` 代码行索引由 `codeline@index` 开关控制。

`\CodelineNumbered`

```
836 \newif\ifcodeline@index \codeline@indexfalse
837 \let\CodelineNumbered\codeline@indextrue
```

`\codeline@wrindex` 代码索引条目由 `\special@index` 写出。如果按代码行索引，这将被 `\let` 为 `\codeline@wrindex`；如果按页索引，这只是 `\index`。然而，如果给出了 `\nofiles`，我们完全省略写出这样的索引条目。

```
838 \def\codeline@wrindex#1{\if@files
839     \begingroup
840     \set@display@protect
841     \immediate\write\@indexfile
842         {\string\indexentry{#1}%
843         {\number\c@CodelineNo}}%
844     \endgroup
845     \fi}
```

`\special@index` 默认情况下不会写出索引条目。

```
846 \let\special@index = \@gobble
```

`\CodelineIndex` 此命令用于打开使用 `\makeindex` 的索引文件，设置开关以指示代码行编号，并相应地定义 `\special@index`。

```
847 \def\CodelineIndex{\makeindex
848     \codeline@indextrue
849     \let\special@index\codeline@wrindex}
```

`\PageIndex` `\PageIndex` 类似。

```
850 \def\PageIndex{\makeindex
851     \codeline@indexfalse
852     \let\special@index\index}
```

`CodelineNo` (*counter*) 我们需要一个计数器来跟踪行号。

```
853 \newcount\c@CodelineNo \c@CodelineNo\z@
```

²⁷提示：本评论由 Dave Love 撰写。

`\theCodelineNo` 这提供了一个钩子来控制行号的格式，可以在类文件中定义。

```
854 \ifundefined{theCodelineNo}
855   {\ifx\selectfont\undefined
856     \def\theCodelineNo{\rmfamily\scriptsize\arabic{CodelineNo}}}%
857   \else
858     \def\theCodelineNo{\reset@font\scriptsize\arabic{CodelineNo}}}%
859   \fi}
860 {}
```

7.15 用于文档化包文件的布局参数

`\tolerance` 文档化包文件的人可能更愿意在过满的 `\hbox` 中使事物“突出”，并且间距可能不太好，因为他们可能不想花费太多时间使所有换行都完美无缺！

```
861 \tolerance=1000\relax
```

以下的 `\mathcode` 定义允许字符 ‘\’ 和 ‘@’ 在数学模式中以 `\ttfamily` 字体显示；²⁸ 特别适用于像 `\@abc=1` 这样的情况。

如果正在使用旧的 `german` 宏包版本，那么 ‘”’ 字符是活动的，并且可能会影响下面的 *〈16-bit number〉* 数量的定义，因此我们在一个组内更改了 “ 的 `\catcode`，并使用了 `\global`。

```
862 { \catcode'\="=12
863 \global\mathcode'\="705C \global\mathcode'\@="7040 }
```

`\DocstyleParms` 这个宏可以用来为 `\MacrocodeTopsep` 和 `\MacroIndent` 以及一些其他内部寄存器分配新值。如果已经定义了该宏，将不会执行默认定义。请注意，如果应该在类文件中（例如像 `ltugboat.cls`）分配新值，则需要通过该宏进行，因为在读取 `doc.sty` 之前，这些寄存器是未定义的。内部寄存器的默认值分散在此文件中。

```
864 \ifundefined{DocstyleParms}{\DocstyleParms}
```

使用或未使用后清除 `\DocstyleParms`。

```
865 \let\DocstyleParms\relax
```

`\AmSTeX` 这里有一些定义，在文档化包文件时可以有用地使用：现在我们可以轻松地引

`\BibTeX` 用 \mathcal{A} 、 \mathcal{B} 、 \mathcal{C} 、 \mathcal{D} 、 \mathcal{E} 、 \mathcal{F} 、 \mathcal{G} 、 \mathcal{H} 、 \mathcal{I} 、 \mathcal{J} 、 \mathcal{K} 、 \mathcal{L} 、 \mathcal{M} 、 \mathcal{N} 、 \mathcal{O} 、 \mathcal{P} 、 \mathcal{Q} 、 \mathcal{R} 、 \mathcal{S} 、 \mathcal{T} 、 \mathcal{U} 、 \mathcal{V} 、 \mathcal{W} 、 \mathcal{X} 、 \mathcal{Y} 、 \mathcal{Z} ，以及通常的 \mathcal{A} 、 \mathcal{B} 、 \mathcal{C} 、 \mathcal{D} 、 \mathcal{E} 、 \mathcal{F} 、 \mathcal{G} 、 \mathcal{H} 、 \mathcal{I} 、 \mathcal{J} 、 \mathcal{K} 、 \mathcal{L} 、 \mathcal{M} 、 \mathcal{N} 、 \mathcal{O} 、 \mathcal{P} 、 \mathcal{Q} 、 \mathcal{R} 、 \mathcal{S} 、 \mathcal{T} 、 \mathcal{U} 、 \mathcal{V} 、 \mathcal{W} 、 \mathcal{X} 、 \mathcal{Y} 、 \mathcal{Z} 。

```
\SliTeX 866 \ifundefined{AmSTeX}
867   {\def\AmSTeX{\leavevmode\hbox{$\mathcal{A}\kern-.2em\lower.376ex%
```

²⁸你可能会想为什么这些定义声称两个字符都属于变量族（即前面的数字 7）。原因在于：最初，`\mathcode` 中 `\` 的定义为 “075C”，即数字族号为 7 中的普通字符编号 92（十六进制为 5C），这是标准 \mathcal{A} 中的打字机族。但是这个文件不应该依赖于这个特定的设置，因此我将这些 `\mathcodes` 修改为适用于任何族的分配。举个例子，请参阅关于新字体选择方案的文章。

```

868      \hbox{$\mathcal M$}\kern-.2em\mathcal S$-\TeX}}{}{}
869 \@ifundefined{BibTeX}
870   {\def\BibTeX{{\rmfamily B\kern-.05em%
871     \textsc{i\kern-.025em b}\kern-.08em%
872     T\kern-.1667em\lower.7ex\hbox{E}\kern-.125emX}}}{}}
873 \@ifundefined{SliTeX}
874   {\def\SliTeX{{\rmfamily S\kern-.06emL\kern-.18em\raise.32ex\hbox
875     {\scshape i}\kern-.03em\TeX}}}{}}

```

\PlainTeX 这里甚至有一个 PLAIN T_EX 和一个 WEB。

```

\Web 876 \@ifundefined{PlainTeX}{\def\PlainTeX{\textsc{Plain}\kern2pt\TeX}}{}
877 \@ifundefined{Web}{\def\Web{\textsc{Web}}}{}}

```

7.16 更改 % 的 \catcode

\MakePercentIgnore 最后最重要的部分：我们改变 ‘%’ 的 \catcode，使其被忽略（这就是我们能够生成此文档的方式！）。我们提供了两个命令来执行实际的切换。

```

878 \def\MakePercentIgnore{\catcode'\%9\relax}
879 \def\MakePercentComment{\catcode'\%14\relax}

```

\DocInput 上面的两个宏现在被用来定义 \DocInput 宏，该宏在 doc 包的 v1.5l（或类似版本）中引入。在旧版本中，\MakePercentIgnore 被放置在 doc.sty 的最后。

```

880 \def\DocInput#1{\MakePercentIgnore\input{#1}\MakePercentComment}

```

7.17 GetFileInfo

\GetFileInfo 根据 \ProvidesPackage 等命令中的信息定义 \filedate 和相关命令。

```

881 \def\GetFileInfo#1{%
882   \def\filename{#1}%
883   \def\@tempb##1 ##2 ##3\relax##4\relax{%
884     \def\filedate{##1}%
885     \def\fileversion{##2}%
886     \def\fileinfo{##3}}%
887   \edef\@tempa{\csname ver@#1\endcsname}%
888   \expandafter\@tempb\@tempa\relax ? \relax\relax}

```

8 整合 hypdoc

如果选择了 hyperref 选项（这是默认选项），我们加载 hypdoc 包。我们尽可能晚地加载它，以防止如果在导言区也加载它，会产生选项冲突。当前该

包更改了比它应该更改的更多命令（不知道下面定义的新定义），所以我们必须保存和恢复一些。

中期内，hypdoc 中的所有代码应直接包含在 doc 中。现在，当它们是分开的时候，我们必须做这些调整。

```
889 \AddToHook{begindocument/before}[doc/hyperref]{%
890 \ifdoc@hyperref
```

围绕问题 #22 编写代码很烦人

```
891 \expandafter\let\expandafter\doc@eoph@@k\csname doc.sty-h@@k\endcsname
```

我们需要无任何选项加载包，因此如果已经加载，就不会有选项冲突。

```
892 \RequirePackage{hypdoc}
893 \expandafter\let\csname doc.sty-h@@k\endcsname\doc@eoph@@k
```

在加载 hypdoc 之后，我们需要再次取消定义这些宏，这样稍后 Macro 和 Env doc 项看起来是未定义的。

```
894 \let\PrintDescribeMacro \@@PrintDescribeMacro
895 \let\PrintDescribeEnv \@@PrintDescribeEnv
896 \let\PrintMacroName \@@PrintMacroName
897 \let\PrintEnvName \@@PrintEnvName
898 \let\SpecialUsageIndex \@@SpecialUsageIndex
899 \let\SpecialEnvIndex \@@SpecialEnvIndex
900 \let\SortIndex \@@SortIndex
901 \let\DescribeMacro \@@DescribeMacro
902 \let\DescribeEnv \@@DescribeEnv
```

该包将新的 \special@index 定义添加到 \CodelineIndex 和 \PageIndex 中，但由于我们加载它得很晚，我们已经超过了它们（在导言区）。因此，我们在这里测试最终状态，如果有必要的话，在这里进行。

```
903 \ifx\special@index\@gobble % do we write index entries at all?
904 \else
905   \ifcodeline@index
906     \let\special@index\HD@codeline@wrindex
907   \else
908     \let\special@index\HD@page@wrindex
909   \fi
910 \fi
```

amsmath 文档在标题和启用 hyperref 时使用 \env，这在书签中会引起问题。

TODO: 最终在其他地方进行修复

```
911 \AddToHook{class/amstx/after}{%
912   \pdfstringdefDisableCommands{\let\env\@empty }}%
```

该包还在 `\index` 条目中添加了额外的代码，但它不知道 `doc` 现在所做的所有事情。因此，我们需要提供两个帮助程序来处理某些条目中的 `\encapchar` 情况。

```
913 \def\doc@providetarget{\HD@target}%
914 \def\doc@handleencap#1{\encapchar hdclindex{\the\c@HD@hypercount}{#1}}%
```

如果未加载该包，这些辅助函数几乎无效。

```
915 \else
916 \let\doc@providetarget\@empty
917 \def\doc@handleencap#1{\encapchar #1}%
```

我们定义下一个命令，以防用户将选项 `hyperref` 从 `true` 更改为 `false`，但未删除辅助文件。

```
918 \def\hdclindex#1#2{\ifx\@nil#2\@nil\else\csname #2\expandafter\endcsname\fi}%
919 \def\hdpindex #1{\ifx\@nil#1\@nil\else\csname #1\expandafter\endcsname\fi}%
920 \fi
921 }
```

9 集成 DoX 包代码

这一部分的代码大部分都是从 DoX 包中借用的，由 Didier 制作，只做了一些小的修改（到目前为止）。这意味着需要来回调整代码和文档，并且两者都需要进一步更新。

9.1 DoX 环境

`\@doc@env` **TODO:** 原始文档 – 修复

`\@doc@env@` `{\are-we-macrolike}{\item}{\indextype}{\name}`

在 `doc.sty` 中，`macro` 和 `environment` 环境通过 `\m@cro@` 宏进行处理，该宏通过测试作为其第一个参数的布尔条件来实现特定部分。这种机制不是可扩展的，因此我必须手工制作一个更通用的版本，适用于任何新的 `dox` 项目，看起来与原始版本非常相似（添加了选项管理）。

第一步是检查 #3 中是否有一个逗号分隔的名称列表，如果有，我们就调用单独为每个名称执行工作的宏。

```
922 \ExplSyntaxOn
923 \long\def\@doc@env#1#2#3{
\endgroup 在这里关闭名称的扫描（使用特殊类别码）。
924 \endgroup
925 \clist_map_inline:nn {#3} { \@doc@env@{#1}{#2}{##1} }
```

```

926 }
927
928 \ExplSyntaxOff

```

每个给定列表中名称的有效载荷如下：

```

929 \long\def\@doc@env@#1#2#3{%
930   \topsep\MacroTopsep
931   \trivlist
932   \edef\saved@macroname{\string#3}%

```

自版本 2.1g 开始，doc 创建了一个 `\saved@indexname` 命令，由 `\changes` 使用。我们现在也支持这个。该命令的扩展取决于所述项是否类似于宏，这里我们不知道（只有 `\NewDocElement` 知道）。这就是为什么我们需要为每个单独的项生成 `\saved@indexname` 的正确方式的一个特定命令。这些命令命名为 `\@Save<item>IndexName`；它们在技术上是生成的 API 的一部分，只是不用于公共使用。

TODO: 上面的文档已不再正确（但代码仍需要进一步更改）

#1 要么是 TT（表示真 = 类似宏），要么是 TF。如果是真，我们将从 `\saved@macroname` 中删除第一个字符，将结果存储在 `\saved@indexname` 中，并在索引中使用后者进行排序。

```

933   \if #1%
934     \edef\saved@indexname{\expandafter\@gobble\saved@macroname}%
935 %

```

如果描述的 doc 元素类似于宏但不是普通的“宏”，则应记录其类型，这是发生这种情况的地方。对于宏（应该占据这些项的大部分），我们不这样做，对于从索引的角度看起来像宏的任何其他内容，我们也不这样做，以保持异常项的列表较小。这将是索引命令 `\Code<doc-element>Index` 等同于 `\CodeMacroIndex` 的情况。

```

936   \expandafter\ifx
937     \csname Code#2Index\endcsname
938     \CodeMacroIndex
939   \else
940     \record@index@type@save
941     {\saved@indexname}{#2}%
942   \fi
943 \else
944   \let\saved@indexname\saved@macroname
945 \fi
946 %
947 \def\makelabel##1{\llap{##1}}%
948 \if@inlabel

```

```

949     \let\@tempa\@empty
950     \count@\macro@cnt
951     \loop\ifnum\count@>\z@
952         \edef\@tempa{\@tempa\hbox{\strut}}\advance\count@\m@ne
953     \repeat
954     \edef\makelabel##1{\llap{\vtop to\baselineskip{\@tempa\hbox{##1}\vss}}}%
955     \advance\macro@cnt\@ne
956 \else
957     \macro@cnt\@ne
958 \fi
959 \ifdoc@noprint
960     \item
961 \else
962     \edef\@tempa{%
963         \noexpand\item[%

```

对原始宏的第二个显著修改涉及动态构建打印宏的名称:

```

964         \noexpand\doc@providetarget
965         \noexpand\strut
966         \noexpand\@nameuse{Print#2Name}{\saved@macroname}}}%
967     \@tempa
968 \fi
969 \ifdoc@noindex\else
970     \global\advance\c@CodelineNo\@ne

```

第三个修改涉及动态构建索引宏的名称:

```

971     \csname SpecialMain#2Index\expandafter\endcsname
972     \expandafter{\saved@macroname}\nobreak
973     \global\advance\c@CodelineNo\m@ne
974 \fi

```

在类似宏环境中抑制进一步的`\index` 条目。(对于非类似宏环境来说这样做没有意义, 因为索引条目仅针对以反斜杠开头的项生成。)

TODO: 修正

```

975     \if#1\expandafter\DoNotIndex \expandafter {\saved@macroname}\fi
976     \ignorespaces}

```

`\doc@env` $\{\langle true-value \rangle\}\{\langle item \rangle\}[\langle options \rangle]$

处理可选参数并调用 `\doc@env`。因为环境可以嵌套, 我们不能依赖分组来获取选项的默认值。因此, 我们需要在每次调用时重置选项。

TODO: 如果可用, 使用 `2e` 接口的 `\keys_set:nn`

```

977 \def\doc@env#1#2[#3]{%
978     \@nameuse{doc@noprint\doc@noprintdefault}%

```



```

979 \@nameuse{doc@noindex\doc@noindexdefault}%
980 \csname keys_set:nn\endcsname{doc}{#3}%
981 \begingroup
982   \ifdoc@outer
983     \catcode'\12
984   \fi
985   \MakePrivateLetters
986   \@doc@env{#1}{#2}%
987 }

```

9.2 doc 描述

`\@doc@describe {< 项>}{< 名称>}`

```

988 \def\@doc@describe#1#2{%
989   \ifdoc@noprint\else
990     \marginpar{\raggedleft

```

hyperref 目标必须处于水平模式（如果在 `\strut` 之后，则是这种情况）。

```

991       \strut
992       \doc@providetarget
993       \@nameuse{PrintDescribe#1}{#2}}%
994   \fi
995   \ifdoc@noindex\else
996     \@nameuse{Special#1Index}{#2}%
997   \fi
998   \@esphack
999 \endgroup
1000 \ignorespaces}

```

`\doc@describe {< 项>}[< 选项>]`

处理可选参数并调用 `\@doc@describe`。

TODO: 如果可用，使用 *2e* 接口的 `\keys_set:nn`

```

1001 \def\doc@describe#1[#2]{%
1002   \leavevmode\@bsphack
1003   \csname keys_set:nn\endcsname{doc}{#2}%
1004   \@doc@describe{#1}}

```

9.3 API 构建

`\@temptokenb` 一个临时寄存器（可能在其他地方定义过）

```

1005 \@ifundefined{temptokenb}{\newtoks\@temptokenb}{}

```

```

\doc@createspecialmainindex {\item}{\idxtype}{\idxcat}

createspecialmainmacrolikeindex {\item}{\idxtype}{\idxcat}
\doc@createspecialmainmacrolikeindex
    TODO: 原始文档 – 修正
    “类似宏” 版本类似于 doc 的 \SpecialIndex@ 宏，但更简化。希望永远
    不会有人将 \_ 或非字母宏定义为类似宏的 doc 元素...

1006 \def\doc@createspecialindexes#1#2#3{%
1007     \@temptokena{\space (#2)}%
1008     \@temptokenb{#3:}%
1009     \@namedef{SpecialMain#1Index}##1{%
1010         \noexpand\@bsphack
1011         \ifdoc@toplevel
1012             \noexpand\special@index{##1\noexpand\actualchar
1013                 {\string\ttfamily\space##1}%
1014                 \ifx\@nil#2\@nil\else \the\@temptokena \fi
1015                 \noexpand\encapchar main}%
1016         \fi
1017         \ifx\@nil#3\@nil\else
1018             \noexpand\special@index{\the\@temptokenb\noexpand\levelchar
1019                 ##1\noexpand\actualchar{\string\ttfamily\space##1}%
1020                 \noexpand\encapchar main}%
1021         \fi
1022         \noexpand\@esphack}%
1023     \@namedef{Special#1Index}##1{%
1024         \noexpand\@bsphack
1025         \ifdoc@toplevel
1026             \noexpand\doc@providetarget
1027             \noexpand\index{##1\noexpand\actualchar{\string\ttfamily\space##1}%
1028                 \ifx\@nil#2\@nil\else \the\@temptokena \fi
1029                 \noexpand\doc@handleencap{usage}}%
1030         \fi
1031         \ifx\@nil#3\@nil\else
1032             \noexpand\index{\the\@temptokenb\noexpand\levelchar
1033                 ##1\noexpand\actualchar{\string\ttfamily\space##1}%
1034                 \noexpand\doc@handleencap{usage}}%
1035         \fi
1036         \noexpand\@esphack}}
1037 \def\doc@createspecialmacrolikeindexes#1#2#3{%
1038     \@temptokena{\space (#2)}%
1039     \@temptokenb{#3:}%

```

```

1040 \nameedef{Code#1Index}##1##2{%
1041   \noexpand\@SpecialIndexHelper@##2\noexpand\@nil
1042   \noexpand\@bsphack
1043   \noexpand\ifdoc@noindex\noexpand\else
1044     \ifdoc@toplevel
1045       \noexpand\special@index{\noexpand\@gtempa\noexpand\actualchar
1046 \string\verb% % 为了糊弄 emacs 的代码高亮
1047 \noexpand\quotechar*\noexpand\verbatimchar
1048 \noexpand\bslash\noexpand\@gtempa\noexpand\verbatimchar
1049 \ifx\@nil#2\@nil\else \the\@temptokena \fi
1050 \noexpand\encapchar ##1}%
1051   \fi
1052   \ifx\@nil#3\@nil\else
1053     \noexpand\special@index{\the\@temptokenb\noexpand\levelchar
1054 \noexpand\@gtempa\noexpand\actualchar
1055 \string\verb% % 为了糊弄 emacs 的代码高亮
1056 \noexpand\quotechar*\noexpand\verbatimchar
1057 \noexpand\bslash\noexpand\@gtempa\noexpand\verbatimchar
1058 \noexpand\encapchar ##1}%
1059   \fi
1060 \noexpand\fi
1061 \noexpand\@esphack}%

1062 \nameedef{SpecialMain#1Index}##1{%
1063   \expandafter\noexpand\csname Code#1Index\endcsname
1064     {main}{##1}}%

1065 \nameedef{Special#1Index}##1{%
1066   \noexpand\@SpecialIndexHelper@##1\noexpand\@nil
1067   \noexpand\@bsphack
1068   \noexpand\ifdoc@noindex\noexpand\else
1069     \ifdoc@toplevel
1070       \noexpand\doc@providetarget
1071       \noexpand\index{\noexpand\@gtempa\noexpand\actualchar
1072 \string\verb% % 为了糊弄 emacs 的代码高亮
1073 \noexpand\quotechar*\noexpand\verbatimchar
1074 \noexpand\bslash\noexpand\@gtempa\noexpand\verbatimchar
1075 \ifx\@nil#2\@nil\else \the\@temptokena \fi
1076 \noexpand\doc@handleencap{usage}}%
1077   \fi
1078   \ifx\@nil#3\@nil\else
1079     \noexpand\index{\the\@temptokenb\noexpand\levelchar
1080 \noexpand\@gtempa\noexpand\actualchar
1081 \string\verb% % 为了糊弄 emacs 的代码高亮

```

```

1082 \noexpand\quotechar*\noexpand\verbatimchar
1083 \noexpand\bslash\noexpand\@gtempa\noexpand\verbatimchar
1084 \noexpand\doc@handleencap{usage}}%
1085     \fi
1086 \noexpand\fi
1087     \noexpand\@esphack}}

```

`\doc@createdescribe {<item>}`

```

1088 \def\doc@createdescribe#1{%
1089     \@namedef{Describe#1}{%

```

由于可选参数，我们必须在解析之前设置 `\MakePrivateLetters`（抱歉冒昧猜测）。否则，不正确但相当常见的用法，比如 `\DescribeMacro\foo@bar`，将会出现问题，因为对可选参数的扫描将标记后续输入（即，在这种情况下是 `\foo`），在 `@` 符号变成字母之前。结果是 `DescribeMacro` 只会接收到 `\foo` 作为它的参数。

```

1090     \begingroup
1091     \MakePrivateLetters
1092     \@ifnextchar[%]
1093     {\doc@describe{#1}}{\doc@describe{#1}[]}}

```

`\doc@createenv {<item>}{<envname>}`

```

1094 \def\doc@createenv#1#2#3{%
1095     \@namedef{#3}{%
1096         \@ifnextchar[%]
1097         {\doc@env{#1}{#2}}{\doc@env{#1}{#2}[]}}%

```

而不是将环境的结束设 (`\let`) 为 `\endtrivlist`，我们使用一级扩展。这样，如果该环境发生任何可能的更改（如果真的发生了），都会得到正确的反映。

```

1098     \@namedef{end#3}{\endtrivlist}%
1099 % \expandafter\let\csname end#3\endcsname\endtrivlist
1100 }

```

`\@nameedef`

```

1101 \def\@nameedef#1{\expandafter\edef\csname #1\endcsname}

```

9.4 API 创建

整个用户界面在一个宏调用中创建。

defaults:

```
idxtype = #3
```

```

idxgroup = #3s
printtype =

\doc@declareerror

1102 \def\doc@declareerror#1#2{%
1103   \PackageError{doc}{Doc element '#1/#2' already defined?\@gobble}%
1104   {There is already a definition for
1105     '\string\Print#1Name',\MessageBreak
1106     '\string\PrintDescribe#1'
1107     or the environment '#2'.\MessageBreak
1108     Maybe you are overwriting something by mistake!\MessageBreak
1109     Otherwise use '\string\RenewDocElement' instead.}%
1110 }

```

```

\doc@notdeclarederror

1111 \def\doc@notdeclarederror#1#2{%
1112   \PackageError{doc}{Doc element '#1/#2' unknown}%
1113   {I expected an existing definition for
1114     '\string\Print#1Name',\MessageBreak
1115     '\string\PrintDescribe#1' and
1116     the environment '#2' but\MessageBreak
1117     not all of them are defined.\MessageBreak
1118     Maybe you wanted to use
1119     '\string\NewDocElement'?\}%
1120 }

```

```

\NewDocElement [<options>]{<name>}{<envname>}

1121 \newcommand\NewDocElement[3][{}]{%
1122   \@ifundefined{Print#2Name}%
1123   {\@ifundefined{PrintDescribe#2}%
1124     {\@ifundefined{#3}%
1125       {\@ifundefined{end#3}%
1126         {\@NewDocElement{#1}}%
1127         \doc@declareerror
1128       }\doc@declareerror
1129     }\doc@declareerror
1130   }\doc@declareerror
1131   {#2}{#3}%
1132 }

```

```

\RenewDocElement [<options>]{<name>}{<envname>}

1133 \newcommand\RenewDocElement[3][{}]{%

```

```

1134 \ifundefined{Print#2Name}\doc@notdeclarederror
1135     {\ifundefined{PrintDescribe#2}\doc@notdeclarederror
1136         {\ifundefined{#3}\doc@notdeclarederror
1137             {\ifundefined{end#3}\doc@notdeclarederror
1138                 {\@NewDocElement{#1}}}%
1139             }%
1140         }%
1141     }%
1142 {#2}{#3}%
1143 }

\@NewDocElement {\langle options\rangle}{\langle name\rangle}{\langle envname\rangle}

1144 \def\@NewDocElement#1#2#3{%
1145     \doc@macrolikefalse
1146     \doc@topleveltrue

    TODO: 如果可用，使用 2e 接口的 \keys_set:nn

1147     \def\doc@idxtype{#3}%
1148     \def\doc@idxgroup{#3s}%
1149     \let\doc@printtype\@empty
1150     \csname keys_set:nn\endcsname{doc}{#1}%

\Print...Name {\langle name\rangle}

    TODO: 这么多的 \expandafter，非常混乱……应重新用 expl3 实现

1151     \ifx\doc@printtype\@empty
1152         \@temptokena{}%
1153     \else
1154         \@temptokena\expandafter{\expandafter
1155             \textnormal\expandafter{\expandafter
1156                 \space\expandafter
1157                 (\doc@printtype)}}%
1158     \fi
1159     \@nameedef{Print#2Name}##1{%
1160         {\noexpand\MacroFont
1161             \ifdoc@macrolike
1162                 \noexpand\string
1163             \fi
1164             ##1%
1165             \the\@temptokena
1166         }}%

```

```

\PrintDescribe... {\langle name\rangle}

1167 \expandafter\let\csname PrintDescribe#2\expandafter\endcsname
1168 \csname Print#2Name\endcsname

\SpecialMain...Index {\langle name\rangle}

\Special...Index {\langle name\rangle}

1169 \edef\doc@expr{%
1170 \ifdoc@macrolike
1171 \noexpand\doc@createspecialmacrolikeindexes
1172 \else
1173 \noexpand\doc@createspecialindexes
1174 \fi
1175 {\#2}%
1176 }%
1177 \expandafter\expandafter\expandafter
1178 \doc@expr
1179 \expandafter\expandafter\expandafter
1180 {\expandafter\doc@idxtype\expandafter}\expandafter
1181 {\doc@idxgroup}%

\Describe... [\langle options\rangle]{\langle name\rangle}

1182 \doc@createdescribe{\#2}%

\metaDocElement (env.) TODO: 参数中不能有格式 - 修复
[\langle options\rangle]{\langle name\rangle}

1183 \ifdoc@macrolike
1184 \doc@createenv{TT}{\#2}{\#3}%
1185 \else
1186 \doc@createenv{TF}{\#2}{\#3}%
1187 \fi
1188 }

```

9.5 设置默认的 doc 元素

9.5.1 宏设施

宏只得到一个索引条目（没有索引组，没有索引类型），在边注中打印时也不会得到任何标签。

```

1189 \NewDocElement[macrolike = true ,
1190 idxtype = ,

```

```

1191             idxgroup = ,
1192             printtype =
1193             ]{Macro}{macro}

```

`SpecialMainIndex` 在 doc v2 中, 我们有 `\SpecialMainIndex` 和 `\SpecialMainEnvIndex`, 但现在有了额外的 doc 元素后, 我们总是在“Main”后面添加元素名称, 因此这将是 `\SpecialMainMacroIndex`。我们使用 `\def` 而不是 `\let`, 这样 `\SpecialMainMacroIndex` 的任何重新定义都将是透明的。

```

1194 \def\SpecialMainIndex{\SpecialMainMacroIndex}

```

`SpecialUsageIndex` doc v2 同样有 `\SpecialUsageIndex`, 现在称为 `\SpecialMacroIndex`, 为宏生成“usage”索引条目。同样, 我们通过 `\def` 提供它作为别名。

实际上, doc v2 的文档声称可以同时用于宏和环境, 但事实并非如此, 对于环境, 结果是索引排序中会删除第一个字符。正确的方式是使用 `\SpecialEnvIndex`。

```

1195 \def\SpecialUsageIndex{\SpecialMacroIndex}

```

`\SpecialIndex`

```

1196 \def\SpecialIndex      {\CodeMacroIndex{code}}

```

9.5.2 环境设施

为环境提供文档支持。在打印名称时, 我们与 doc V2 有所不同, 使用“(env.)”标记环境。

```

1197 \NewDocElement[macrolike = false ,
1198             idxtype   = env. ,
1199             idxgroup   = environments ,
1200             printtype = \textit{env.}
1201             ]{Env}{environment}

```

为了在加载 hypdoc 后能够恢复定义, 最好在这里保存它们。我们只在导言区的末尾加载包, 但用户可能会提前加载, 这样会导致混乱。

```

1202 \let\@@PrintDescribeMacro \PrintDescribeMacro
1203 \let\@@PrintDescribeEnv \PrintDescribeEnv
1204 \let\@@PrintMacroName \PrintMacroName
1205 \let\@@PrintEnvName \PrintEnvName
1206 \let\@@SpecialUsageIndex \SpecialUsageIndex
1207 \let\@@SpecialEnvIndex \SpecialEnvIndex
1208 \let\@@SortIndex \SortIndex
1209 \let\@@DescribeMacro \DescribeMacro
1210 \let\@@DescribeEnv \DescribeEnv

```


10 杂项新增

`\cs`

```
1211 \DeclareRobustCommand\cs[1]{\texttt{\backslash #1}}
```

`amstex` 对 `\cs` 有自己的定义，但现在会被覆盖，因为该类在之后加载了 `doc`。所以暂时我们在这里重新安装它。

TODO: 在其他地方修复

```
1212 \AddToHook{class/amstex/after}{%
1213   \DeclareRobustCommand\cs[1]{%
1214     \@boxorbreak{%
1215       \ntt
1216       \addbackslash#1\@empty
1217       \@xp\@xp\@xp\@indexcs\@xp\@nobsbackslash\string#1\@nil
1218     }%
1219   }%
1220   \def\cn{\cs}%
1221 }
```

现在我们可以完成 `docstrip` 的主模块。

```
1222 \</package>
```

参考文献

- [1] G. A. BÜRGER. Wunderbare Reisen zu Wasser und zu Lande, Feldzüge und lustige Abenteuer des Freyherrn v. Münchhausen. London, 1786 & 1788.
- [2] D. E. KNUTH. Literate Programming. *Computer Journal*, Vol. 27, pp. 97–111, May 1984.
- [3] D. E. KNUTH. Computers & Typesetting (The \TeX book). Addison-Wesley, Vol. A, 1986.
- [4] L. LAMPORT. MakeIndex: An Index Processor for \LaTeX . 17 February 1987. (Taken from the file `makeindex.tex` provided with the program source code.)
- [5] FRANK MITTELBACH. The `doc`-option. *TUGboat*, Vol. 10(2), pp. 245–273, July 1989.

- [6] FRANK MITTELBACH, DENYS DUCHIER AND JOHANNES BRAAMS. `docstrip.dtx`. The file is part of core L^AT_EX.
- [7] R. E. RASPE (*1737, †1797). Baron Münchhausens narrative of his marvelous travels and campaigns in Russia. Oxford, 1785.
- [8] RAINER SCHÖPF. A New Implementation of L^AT_EX's `verbatim` and `verbatim*` Environments. File `verbatim.doc`, version 1.4i.

索引

斜体数字指向相应条目描述的页面；下划线数字指向定义的代码行；罗马数字指向代码行。

Symbols

`\-` *l-198, l-826*
`\^` *l-392, l-830*
`^^A` *5, l-24*
`^^X` *5, l-24*

L

L^AT_EX 命令:

`\@@DescribeEnv` *l-902, l-1210*
`\@@DescribeMacro` *l-901, l-1209*
`\@@PrintDescribeEnv` . *l-895, l-1203*
`\@@PrintDescribeMacro` *l-894, l-1202*
`\@@PrintEnvName` *l-897, l-1205*
`\@@PrintMacroName` ... *l-896, l-1204*
`\@@SortIndex` *l-900, l-1208*
`\@@SpecialEnvIndex` .. *l-899, l-1207*
`\@@SpecialUsageIndex` *l-898, l-1206*
`\@MakeShortVerb` *l-711, l-712, l-713*
`\@NewDocElement`
 *l-1126, l-1138, l-1144*
`\@SpecialIndexHelper@`
 *l-387, l-469, l-1041, l-1066*
`\@auxout` *l-326, l-338*
`\@boxorbreak` *l-1214*
`\@doc@describe` *l-988, l-1004*
`\@doc@env` *l-922, l-986*
`\@doc@env@` *l-922*

`\@idxitem`
 .. *l-515, l-521, l-557, l-628, l-635*
`\@ifnextchar` *l-1092, l-1096*
`\@indexcs` *l-1217*
`\@indexfile` *l-841*
`\@input@` *l-582, l-658*
`\@labels` *l-72*
`\@makefntext` *l-697*
`\@minipagefalse` *l-84*
`\@nameedef` *l-1009, l-1023, l-1040,*
 l-1062, l-1065, l-1101, l-1159
`\@nameuse`
 .. *l-966, l-978, l-979, l-993, l-996*
`\@newlistfalse` *l-83*
`\@nobs slash` *l-1217*
`\@restonecolfalse` ... *l-518, l-632*
`\@restonecoltrue` ... *l-518, l-632*
`\@setupverbvisiblespace` ... *l-221*
`\@shortvr bdef`
 *l-711, l-712, l-715, l-724*
`\@shortvr binfo`
 .. *l-715, l-728, l-734, l-738, l-752*
`\@sxverbatim` *l-222*
`\@temptokena` *l-1007,*
 l-1014, l-1028, l-1038, l-1049,
 l-1075, l-1152, l-1154, l-1165

<code>\@temptokenb</code>	<code>\c@GlossaryColumns</code> ..
..... l-1005 , l-1008 , l-1018 ,	<code>\c@HD@hypercount</code>
l-1032 , l-1039 , l-1053 , l-1079	<code>\c@IndexColumns</code>
<code>\@verbatim</code>	l-510 , l-514
l-223	<code>\c@StandardModuleDepth</code>
<code>\@xp</code> l-175 , l-180 , l-187
l-1217	<code>\c_left_brace_str</code> l-398 , l-399 , l-405
<code>__doc_dont_index:n</code>	<code>\c_right_brace_str</code>
..... l-305 , l-308 , l-310 l-401 , l-403 , l-407
<code>__doc_dont_index_aux:n</code>	<code>\ch@angle</code>
..... l-305 , l-313 , l-315	l-146 , l-147
<code>__doc_idxtype_put:Nn</code>	<code>\ch@percent</code>
..... l-324 , l-324 , l-335	l-136 , l-142
<code>__doc_idxtype_put:nn</code>	<code>\ch@plus@etc</code>
.. l-325 , l-330 , l-337 , l-343 , l-343	l-150 , l-152
<code>__doc_idxtype_put_scan:nn</code> ..	<code>\changes@</code>
..... l-336 , l-336 , l-341	l-598 , l-599
<code>__doc_idxtype_put_scan:on</code> ..	<code>\character@table</code>
..... l-341 , l-342	l-801 , l-802
<code>__doc_maybe_index:o</code>	<code>\check@angle</code>
..... l-357 , l-357 , l-361	l-144 , l-146
<code>__doc_maybe_index_aux:Nnn</code> ..	<code>\check@checksum</code>
..... l-376 , l-429 , l-429	l-662 , l-776
<code>__doc_maybe_index_aux:nN</code>	<code>\check@module</code>
..... l-358 , l-363 , l-367 , l-367	l-88 , l-89 , l-133
<code>__doc_maybe_index_short:o</code> ..	<code>\check@modulesfalse</code>
..... l-362 , l-362 , l-366	l-139
<code>__doc_trace:x</code>	<code>\check@modulestrue</code> ..
l-302 , l-312 , l-320 , l-347 ,	l-140 , l-141
l-350 , l-368 , l-371 , l-381 , l-432	<code>\check@percent</code>
<code>\active@escape@char</code>	l-237 , l-242
..... l-51 , l-247 , l-260	<code>\check@plus@etc</code>
<code>\add@special</code>	l-152
l-716 , l-757	<code>\clist</code>
<code>\addbslash</code>	l-925
l-1216	<code>\clist_map_function:nN</code>
<code>\AddToHook</code>	l-313
l-889 , l-911 , l-1212	<code>\close@crossref</code>
<code>\AmSTeX</code>	l-98 , l-265
l-866	<code>\cn</code>
<code>\AtBeginDocument</code>	l-1220
l-28 , l-132	<code>\codeline@index</code>
<code>\AtEndDocument</code>	l-836
l-332	<code>\codeline@indexfalse</code> l-836 , l-851
<code>\BibTeX</code>	<code>\codeline@indextrue</code> .
l-866	l-837 , l-848
<code>\blank@linefalse</code>	<code>\codeline@wrindex</code>
l-76 , l-92	l-838 , l-849
<code>\blank@linetrue</code>	<code>\CodelineIndex</code>
l-78 , l-92	l-847
<code>\box</code>	<code>\CodeMacroIndex</code>
l-72	l-938 , l-1196
<code>\c@CodelineNo</code>	<code>\columnsep</code>
... l-86 , l-843 , l-853 , l-970 , l-973	l-519 , l-551 , l-633
	<code>\columnseprule</code>
	l-519 , l-633
	<code>\cs:w</code>
	l-355
	<code>\cs_end:</code>
	l-355
	<code>\cs_generate_variant:Nn</code>
 l-341 , l-356
	<code>\cs_if_exist:NTF</code>
	l-430
	<code>\cs_new:Npn</code>
	... l-302 , l-305 , l-310 , l-315 ,
	l-324 , l-329 , l-336 , l-343 , l-354 ,
	l-357 , l-362 , l-367 , l-386 , l-429

<code>\cs_set_eq:NN</code>	<code>\doc_dont_index:n</code>
<i>l-333, l-335, l-342, l-361, l-366</i> 41 , l-305 , <i>l-305, l-318</i>
<code>\cs_to_str:N</code>	<code>\DocstyleParms</code> l-864
<code>\DeclareKeys</code>	<code>\documentclass</code> <i>l-2</i>
<code>\DeclareRobustCommand</code>	<code>\DoNotIndex</code> l-318 , <i>l-975</i>
. <i>l-674, l-1211, l-1213</i>	<code>\encodingdefault</code>
<code>\default~table</code> <i>l-102, l-108, l-118, l-124</i>
<code>\DescribeEnv</code> <i>l-902, l-1210</i>	<code>\endmacrocode</code> l-93 , <i>l-204, l-691</i>
<code>\DescribeMacro</code> <i>l-901, l-1209</i>	<code>\endmacrocode*</code> l-203
<code>\do@noligs</code> <i>l-81</i>	<code>\endtheindex</code> l-516
<code>\doc@createdescribe</code> l-1088 , <i>l-1182</i>	<code>\ensuremath</code> <i>l-675, l-685</i>
<code>\doc@createenv</code> l-1094 , <i>l-1184, l-1186</i>	<code>\env</code> <i>l-912</i>
<code>\doc@createspecialindexes</code>	<code>\exp_after:wN</code> <i>l-355</i>
. <i>l-1006, l-1173</i>	<code>\exp_args:co</code> l-354 , <i>l-354</i>
<code>\doc@createspecialmacrolikeindexes</code>	<code>\exp_args:Ncno</code> <i>l-376</i>
. <i>l-1037, l-1171</i>	<code>\exp_args:Nf</code> <i>l-337, l-358</i>
<code>\doc@createspecialmainindex</code>	<code>\exp_args:NNf</code> <i>l-344</i>
. l-1006	<code>\exp_args:No</code> <i>l-363, l-382</i>
<code>\doc@createspecialmainmacrolikeindex</code>	<code>\exp_args:Nx</code> <i>l-325, l-330</i>
. l-1006	<code>\ExplSyntaxOff</code> <i>l-447, l-928</i>
<code>\doc@declareerror</code> l-1102 ,	<code>\ExplSyntaxOn</code> <i>l-299, l-922</i>
<i>l-1127, l-1128, l-1129, l-1130</i>	<code>\filedate</code> <i>l-884</i>
<code>\doc@describe</code> l-1001 , <i>l-1093</i>	<code>\fileinfo</code> <i>l-886</i>
<code>\doc@env</code> l-977 , <i>l-1097</i>	<code>\filename</code> <i>l-882</i>
<code>\doc@eoph@@k</code> <i>l-891, l-893</i>	<code>\fileversion</code> <i>l-885</i>
<code>\doc@expr</code> <i>l-1169, l-1178</i>	<code>\font</code> <i>l-680, l-681</i>
<code>\doc@handleencap</code> <i>l-914</i> ,	<code>\fontencoding</code> <i>l-108, l-124</i>
<i>l-917, l-1029, l-1034, l-1076, l-1084</i>	<code>\fontfamily</code> <i>l-109, l-125</i>
<code>\doc@hyperreftrue</code> <i>l-48</i>	<code>\fontseries</code> <i>l-110, l-126</i>
<code>\doc@idxgroup</code> <i>l-44, l-1148, l-1181</i>	<code>\fontshape</code> <i>l-111, l-127</i>
<code>\doc@idxtype</code> <i>l-43, l-1147, l-1180</i>	<code>\g__doc_idxtype_prop</code>
<code>\doc@macrolikefalse</code> <i>l-1145</i> 41 , l-300 , <i>l-322, l-351, l-374</i>
<code>\doc@multicoltrue</code> <i>l-49</i>	<code>\GetFileInfo</code> l-881
<code>\doc@noindexdefault</code> <i>l-57, l-60, l-979</i>	<code>\glossary@prologue</code>
<code>\doc@noprintdefault</code> <i>l-55, l-978</i> <i>l-627, l-634, l-646</i>
<code>\doc@notdeclarederror</code> l-1111 ,	<code>\group_begin:</code> <i>l-306</i>
<i>l-1134, l-1135, l-1136, l-1137</i>	<code>\group_end:</code> <i>l-311</i>
<code>\doc@printtype</code>	<code>\HD@codeline@wrindex</code> <i>l-906</i>
. <i>l-45, l-1149, l-1151, l-1157</i>	<code>\HD@page@wrindex</code> <i>l-908</i>
<code>\doc@providetarget</code> <i>l-913</i> ,	<code>\HD@target</code> <i>l-913</i>
<i>l-916, l-964, l-992, l-1026, l-1070</i>	<code>\hdclindex</code> <i>l-918</i>
<code>\doc@topleveltrue</code> <i>l-50, l-1146</i>	<code>\hdpindex</code> <i>l-919</i>

<code>\hyphenchar</code>	l-680 , l-681	<code>\macro@finish</code>	l-290 , l-292
<code>\if@compatibility</code> ...	l-100 , l-116	<code>\macro@font</code> l-73 , l-115 , l-181 , l-188	
<code>\if@files</code>	l-838	<code>\macro@name</code>	l-278 , l-286 , l-289
<code>\ifblank@line</code>	l-76 , l-92	<code>\macro@namepart</code> .	l-266 , l-282 ,
<code>\ifcheck@modules</code>	l-134 , l-139		l-283 , l-286 , l-293 , l-295 , l-297
<code>\ifcodeline@index</code>		<code>\macro@switch</code>	l-271 , l-277
....	l-85 , l-537 , l-541 , l-836 , l-905	<code>\macrocode</code>	l-64
<code>\ifdoc@hyperref</code>	l-890	<code>\makeglossary</code>	l-621
<code>\ifdoc@macrolike</code>		<code>\marginparpush</code>	l-214
.....	l-1161 , l-1170 , l-1183	<code>\marginparsep</code>	l-215
<code>\ifdoc@multicol</code>	l-511 , l-624	<code>\marginparwidth</code>	l-214
<code>\ifdoc@noindex</code> ...	l-56 , l-389 ,	<code>\mathsf</code>	l-195
	l-483 , l-969 , l-995 , l-1043 , l-1068	<code>\maybe@index@macro</code> 44 , l-297 , l-357	
<code>\ifdoc@noprint</code> .	l-55 , l-959 , l-989	<code>\maybe@index@short@macro</code>	
<code>\ifdoc@outer</code>	l-982	l-283 , l-362
<code>\ifdoc@reportchangedates</code> ..	l-601	<code>\mddefault</code> l-104 , l-110 , l-120 , l-126	
<code>\ifdoc@toplevel</code>		<code>\MessageBreak</code>	
....	l-1011 , l-1025 , l-1044 , l-1069	..	l-441 , l-443 , l-1105 , l-1107 ,
<code>\ifhmode</code>	l-235		l-1108 , l-1114 , l-1116 , l-1117
<code>\ifnot@excluded</code>	l-294	<code>\meta@font@select</code> ...	l-678 , l-687
<code>\ifpm@module</code>	l-94 , l-133	<code>\meta@hyphen@restore</code> l-679 , l-684	
<code>\ifscan@allowed</code>	l-52 , l-269	<code>\mod@math@codes</code>	l-195 , l-197
<code>\index@prologue</code> l-514 , l-520 , l-530		<code>\Module</code> ..	l-173 , l-178 , l-185 , l-194
<code>\indexentry</code>	l-842	<code>\more@macroname</code>	l-287 , l-288
<code>\indexspace</code>	l-560	<code>\newcommand</code> ..	l-660 , l-1121 , l-1133
<code>\init@checksum</code>	l-663 , l-774	<code>\newcounter</code>	l-192
<code>\init@crossref</code>	l-91 , l-256	<code>\newhelp</code>	l-834
<code>\interlinepenalty</code> l-79 , l-232 , l-235		<code>\newif</code> l-52 , l-92 , l-138 , l-141 , l-836	
<code>\iow_term:x</code>	l-303	<code>\newlanguage</code>	l-672
<code>\it@is@a</code>	l-498 , l-505	<code>\nfss@text</code>	l-676
<code>\itshape</code>	l-687	<code>\noindent</code>	l-697
<code>\l@nohyphenation</code>		<code>\ntt</code>	l-1215
.....	l-227 , l-671 , l-672 , l-682	<code>\PackageError</code>	
<code>\l__doc_donotindex_seq</code>			l-437 , l-789 , l-810 , l-1103 , l-1112
	41 , l-300 , l-316 , l-321 , l-345 , l-369	<code>\PackageInfo</code>	l-753 , l-754
<code>\l__doc_idxtype_tl</code>		<code>\pdfstringdefDisableCommands</code>	
.....	l-374 , l-377 , l-433 , l-438	l-912
<code>\labelsep</code>	l-215	<code>\PlainTeX</code>	l-876
<code>\language</code>	l-227 , l-682	<code>\pm@module</code> l-155 , l-157 , l-165 , l-169	
<code>\legacy_if:nTF</code>	l-303	<code>\pm@modulefalse</code>	l-94 , l-135
<code>\m@th</code>	l-696 , l-698	<code>\pm@moduletrue</code>	l-172
<code>\macro@code</code> .	l-64 , l-67 , l-203 , l-689	<code>\predisplaypenalty</code> l-69 , l-218 , l-220	

<code>\Print</code>	l-1105 , l-1114	<code>\set@display@protect</code>	l-840
<code>\PrintDescribe</code>	l-1106 , l-1115	<code>\shapedefault</code>	l-105 , l-111
<code>\PrintDescribeEnv</code> . . .	l-895 , l-1203	<code>\short@macro</code>	l-279 , l-281
<code>\PrintDescribeMacro</code> .	l-894 , l-1202	<code>\ShowIndexingState</code>	l-319
<code>\PrintEnvName</code>	l-897 , l-1205	<code>\slash@module</code>	l-161 , l-177
<code>\PrintMacroName</code>	l-896 , l-1204	<code>\sldefault</code>	l-121 , l-127
<code>\ProcessKeyOptions</code>	l-51	<code>\SliTeX</code>	l-866
<code>\prop_get:NnNTF</code>	l-374	<code>\special@escape@char</code>	
<code>\prop_gput:Nnn</code>	l-351		l-247 , l-259 , l-267
<code>\prop_new:N</code>	l-301	<code>\special@index</code>	l-418 ,
<code>\prop_show:N</code>	l-322		l-488 , l-494 , l-500 , l-505 , l-846 ,
<code>\protected@edef</code>	l-600		l-849 , l-852 , l-903 , l-906 , l-908 ,
<code>\protected@write</code>	l-326 , l-338		l-1012 , l-1018 , l-1045 , l-1053
<code>\ps@plain</code>	l-706	<code>\SpecialEnvIndex</code>	l-899 , l-1207
<code>\record@index@type@save</code>		<code>\SpecialIndex</code> .	l-295 , l-359 , l-1196
	l-342 , l-940	<code>\SpecialMacroIndex</code>	l-1195
<code>\RecordIndexTypeAux</code> .	l-324 , l-339	<code>\SpecialMainIndex</code> . .	l-1194 , l-1194
<code>\rem@special</code> . . .	l-742 , l-758 , l-763	<code>\SpecialMainMacroIndex</code> . . .	l-1194
<code>\RequirePackage</code>	l-512 , l-892	<code>\SpecialShortIndex</code> . .	l-364 , l-386
<code>\reserved@a</code>	l-392 , l-395 ,	<code>\SpecialUsageIndex</code>	
	l-398 , l-403 , l-409 , l-415 , l-419		l-898 , l-1195 , l-1195 , l-1206
<code>\reserved@b</code>	l-393 , l-396 ,	<code>\star@module</code>	l-159 , l-177
	l-399 , l-407 , l-411 , l-416 , l-422	<code>\step@checksum</code>	l-268 , l-800
<code>\reserved@c</code>	l-394 , l-397 ,	<code>\str_case_e:nnF</code>	l-390
	l-400 , l-408 , l-412 , l-417 , l-424	<code>\subitem</code>	l-557
<code>\reset@font</code>	l-858	<code>\subsubitem</code>	l-557
<code>\reversemarginpar</code>	l-213	<code>\sxmacro@code</code>	l-203 , l-210
<code>\rmfamily</code> .	l-202 , l-856 , l-870 , l-874	<code>\textit</code>	l-580 , l-1200
<code>\saved@indexname</code>		<code>\textnormal</code>	l-1155
	l-609 , l-619 , l-934 , l-941 , l-944	<code>\textsc</code>	l-871 , l-876 , l-877
<code>\saved@macroname</code>		<code>\texttt</code>	l-413 , l-1211
	l-604 , l-613 , l-618 , l-932 ,	<code>\theCodelineNo</code>	l-87 , l-854
	l-934 , l-944 , l-966 , l-972 , l-975	<code>\theindex</code>	l-518
<code>\scan@allowedfalse</code>		<code>\tl_to_str:n</code>	
	l-52 , l-58 , l-274 , l-284		l-312 , l-337 , l-345 , l-356
<code>\scan@allowedtrue</code> l-52 , l-275 , l-285		<code>\tl_to_str:o</code>	l-356 , l-358
<code>\scan@macro</code>	l-260 , l-266	<code>\tl_use:N</code>	l-377 , l-433 , l-438
<code>\scshape</code>	l-875	<code>\tolerance</code>	l-861
<code>\seq_if_in:NnTF</code>	l-345 , l-369	<code>\ttdefault</code> l-103 , l-109 , l-119 , l-125	
<code>\seq_new:N</code>	l-300	<code>\ttfamily</code>	
<code>\seq_put_right:Nx</code>	l-316		l-1013 , l-1019 , l-1027 , l-1033
<code>\seq_show:N</code>	l-321	<code>\unpenalty</code>	l-240

<code>\MakePercentComment</code> . l-878 , l-880	<code>\SpecialEscapechar</code>
<code>\MakePercentIgnore</code> 9 , l-247 , l-262 , l-265
. l-597 , l-878 , l-880	<code>\SpecialIndex</code> 11
<code>\MakePrivateLetters</code> 15 ,	<code>\SpecialMacroIndex</code> 11
l-258 , l-263 , l-307 , l-985 , l-1091	<code>\SpecialMain...Index</code> l-1169
<code>\MakeShortVerb</code> 13 , l-707	<code>\SpecialMainEnvIndex</code> 10
<code>\MakeShortVerb*</code> 13 , l-707	<code>\SpecialMainMacroIndex</code> 10
<code>\maketitle</code> 14 , l-693	<code>\SpecialShortIndex</code> 11
<code>\MaybeStop</code> 13 , l-660 , l-670	<code>\theCodelineNo</code> 10
<code>\meta</code> 13 , l-671	<code>\usage</code> 12 , l-580
<code>\Module</code> 15	<code>\verb</code> 9
<code>\NewDocElement</code> 7 ,	<code>\verbatimchar</code>
l-443 , l-1119 , l-1121 , l-1189 , l-1197	. 11 , l-409 , l-413 , l-421 , l-423 ,
<code>\OnlyDescription</code> l-10 , 13 , l-17 , l-660	l-468 , l-490 , l-491 , l-496 , l-502 ,
<code>\PageIndex</code> 10 , l-850	l-503 , l-507 , l-508 , l-613 , l-614 ,
<code>\percentchar</code>	l-1047 , l-1048 , l-1056 , l-1057 ,
. . l-143 , l-151 , l-163 , l-498 , l-499	l-1073 , l-1074 , l-1082 , l-1083
<code>\PercentIndex</code> l-497 , l-499	宏包命令 (已过时) :
<code>\pfill</code> l-568	<code>\CharacterTable</code> 20 , l-801
<code>\Print...Name</code> l-1151	<code>\CharTableChanges</code> . . . l-801 , l-816
<code>\PrintChanges</code> 15 , l-658	<code>\Checksum</code> 20 , l-799
<code>\PrintDescribe...</code> l-1167	<code>\docdate</code> 24
<code>\PrintDescribeEnv</code> 7	<code>\filedate</code> 24
<code>\PrintDescribeMacro</code> 7	<code>\fileversion</code> 24
<code>\PrintEnvName</code> 7	<code>\OldMakeindex</code> 20 , l-499
<code>\PrintIndex</code> 12 , l-582	<code>\StopEventually</code> 13 , l-670
<code>\PrintMacroName</code> 7	宏包环境:
<code>\ps@titlepage</code> 14 , l-705	<code><DocElement></code> l-1183
<code>\quotechar</code>	environment 7
. 10 , l-409 , l-415 , l-416 , l-421 ,	macro 6
l-458 , l-490 , l-491 , l-496 , l-500 ,	macrocode 5 , l-64
l-502 , l-505 , l-507 , l-508 , l-605 ,	macrocode* 5 , l-203
l-612 , l-1047 , l-1056 , l-1073 , l-1082	theglossary l-624
<code>\RecordChanges</code> l-15 , 15 , l-621	theindex 12 , l-511
<code>\RecordIndexType</code> . . 41 , l-324 , l-440	verbatim 9 , l-218
<code>\RenewDocElement</code> . 8 , l-1109 , l-1133	verbatim* 9 , l-218
<code>\RightBraceIndex</code> l-487	宏包选项:
<code>\SetupDoc</code> 5 , l-16 , l-53 , l-63	debugshow 4
<code>\ShowIndexingState</code> 41	envlike 8
<code>\SortIndex</code> . 11 , l-482 , l-900 , l-1208	hyperref 4
<code>\Special...Index</code> l-1169	idxgroup 8
<code>\SpecialEnvIndex</code> 11	idxtype 8

macrolike	8	T	
multicol	4	TeX 计数器:	
nohyperref	4	\bslash@cnt	l-775 ,
noindex	4		l-781 , l-784 , l-790 , l-797 , l-800
nomulticol	4	\check@sum	l-777 , l-778 ,
noprint	4		l-784 , l-790 , l-796 , l-797 , l-799
notoplevel	8	\guard@level	l-174 , l-175 ,
printtype	8		l-179 , l-180 , l-186 , l-187 , l-193
reportchangedates	4	\hbadness	l-19
toplevel	8	\macro@cnt	l-245 , l-950 , l-955 , l-957
		\tolerance	13

Change History

BHK – 1989/04/26	v1.4? – 1989/04/16
\changes: 改变了 \protect 的定	General: 对索引环境进行更改 ... 50
义。..... 54	v1.4? – 1989/04/19
记录了 \changes 命令。..... 54	General: 使用 DEK 的算法并实现
\glossary@prologue: 添加以支持	双列环境 50
\changes。..... 56	v1.4r – 1989/04/22
GlossaryColumns: 添加以支持	General: 将双列环境放入单独文件 50
\changes。..... 56	v1.4t – 1989/04/24
\GlossaryMin: 添加以支持	\endtheindex: 整合了新的
\changes。..... 56	multicols 环境。..... 51
\GlossaryParms: 添加以支持	IndexColumns: 增加了计数器。... 50
\changes。..... 57	\meta: 添加宏。..... 58
\GlossaryPrologue: 添加以支持	theindex: 整合了新的 multicols 环
\changes。..... 56	境。..... 50
\PrintChanges: 添加以支持	v1.5a – 1989/04/26
\changes。..... 57	General: 现在使用 multicols.sty 而
\RecordChanges: 将原来的	不是 multicols.sty 50
\PrintChanges 命令更名。... 55	theindex: 首先调用 multicols ... 50
\saved@macroname: 为 macro 外排	v1.5c – 1989/04/27
序提供 55	\short@macro: 通过在打印参数之
theglossary: 添加以支持	前放置 scan@allowedfalse 宏,
\changes。..... 56	修复了一个严重错误。..... 39
v1.0p – 1994/05/21	v1.5d – 1989/04/28
General: 使用新的错误命令 1	General: 添加了 \marginparwidth
	设置。..... 34

v1.5f – 1989/04/29		\ps@titlepage: 添加了	
General: 感谢 Brian 记录了		\ps@titlepage	60
\changes 宏的特性。	1	\scan@macro: 增加了对校验和的支	
v1.5g – 1989/05/07		持。	38
General: MacroTopsep 现在称为		\step@checksum: 添加了支持校验	
MacrocodeTopsep, 并添加了新		和的宏。	64
的 MacroTopsep	1	v1.5l – 1989/09/10	
\PlainTeX: plain 和 TeX 之间的空		CodelineNo: 添加计数器以支持代	
格改变。	68	码行号	66
v1.5h – 1989/05/17		\macro@code: 支持代码行号。 . . .	27
General: 所有行缩短至 <72 个字符	1	v1.5m – 1989/09/20	
v1.5i – 1989/06/07		\changes: 第二级别中的	
General: 避免重复读取文件。 . . .	24	\actualchar 被移除。	54
\check@percent: 使用宏\next 来		\CharacterTable: 添加了用于检查	
防止具有参数的宏	35	字符转换问题的宏。	64
将定义更改为 “long”	35	v1.5o – 1989/09/24	
v1.5j – 1989/06/09		\changes: 新的排序。	54
General: 由 Ron Whitney 添加的		v1.5p – 1989/09/28	
修正	1	theglossary: 现在先调用	
\AmSTeX: 宏 AmSTeX 重命名为		\multicols。	56
AmSTeX	67	v1.5q – 1989/11/01	
\maketitle: 移除 thispagestyle		\CharacterTable: 使字符表更易	
plain	59	读。	64
v1.5k – 1989/08/17		v1.5q – 1989/11/03	
\macro@cnt: 解决了保存堆栈问题。	35	General: ‘...Listing macros’ 重命	
v1.5k – 1989/09/04		名为 ‘...Input’。由 R.	
\bslash@cnt: 添加了支持校验和的		Wonneberger 建议	1
宏。	64	v1.5r – 1989/11/04	
\check@checksum: 添加了支持校验		\endmacrocode: 支持代码行号（未	
和的宏。	63	记录）	28
\check@sum: 添加了支持校验和的		macrocode: 支持代码行号（未记录）	26
宏。	64	v1.5s – 1989/11/05	
\CheckSum: 添加了支持校验和的		\codeline@index: 支持代码行号	
宏。	64	（未记录）	66
\Finale: 支持校验和。	57	\it@is@a: 支持代码行号（未记录）	50
\init@checksum: 添加了支持校验		\LeftBraceIndex: 支持代码行号	
和的宏。	63	（未记录）	49
\maketitle: 添加了		\MacroIndent: 支持代码行号（未	
\ps@titlepage	60	记录）	32
\MaybeStop: 支持校验和。	57	\PercentIndex: 支持代码行号（未	
\PrintIndex: 将 \printindex 更		记录）	50
改为 \PrintIndex	53		

\RightBraceIndex: 支持代码行号 (未记录)	49	\scriptsize 前面, 以避免不必要的字体警告。	66
v1.5t – 1989/11/07		\MacroIndent: 将 \rm 移动到 \scriptsize 前面, 以避免不必要的字体警告。	32
\CharacterTable: 使 ~ 在字符表宏 中成为字母。	64	v1.6c – 1990/06/29	
\IndexInput: 调用 \endmacrocode 而不是 \endtrivlist。	59	\changes: 再次新的排序。	54
\macro@code: 添加通用代码。	27	v1.6e – 1991/04/03	
调用 \leavevmode 以获取 \everypar 为空白行。	27	theglossary: 转换为环境定义。	56
macrocode: 通用代码移至 \macro@code。	26	theindex: 转换为环境定义。	50
v1.5u – 1989/11/14		v1.7a – 1992/02/24	
\CharacterTable: 在默认表中将 @ 设置为 “other”。	64	\codeline@index: 记录了代码行号 的支持	66
\check@percent: 添加了等号。	35	v1.7a – 1992/02/25	
\CodelineIndex: 添加了 \PageIndex 和 \CodelineIndex (未记录)	66	General: 修改使用信息	22
\DocstyleParms: \DocStyleParms 现在为空	67	对文本进行了各种小的更改	2
v1.5v – 1990/01/28		\theCodelineNo: 不覆盖现有定义。	67
\changes: “重新编码许多字符”。	54	v1.7a – 1992/02/26	
v1.5w – 1990/02/03		General: 关于宏环境中需要一些文 本的说明。	6
\meta: 允许在空格处断行。	58	在接口部分增加了对 \RecordChanges 等的描述。	15
v1.5w – 1990/02/05		在接口部分文档中记录了 \MakePrivateLetters	15
General: Counter codelineno 重命 名为 CodelineNo	1	文档化了 \verb 的更改。	9
\macro@code: 添加 \@totalleftmargin 的跳过。	27	\@verbatim: 移除了多余的 \tt。	35
v1.5x – 1990/02/17		\bslash: 将 \bslash 的文档移到 “用户接口” 部分	34
\MacroFont: 为 NFSS 添加了 \math@fontsfalse	28	\PrintIndex: 将文档移到接口部分	53
v1.5y – 1990/02/24		v1.7a – 1992/02/27	
CodelineNo: 默认更改。	66	\add@special: 为短引用功能添加。	62
\MacroIndent: 默认更改。	32	\DeleteShortVerb: 添加 (从 newdoc 中移植但现在改变了 \dospecials、\@sanitize)。	61
v1.5z – 1990/04/22		\MakeShortVerb: 新增 (来自 newdoc 但现在修改了 \dospecials 和 \@sanitize)。	60
\Finale: 全局定义 \Finale。	57	\rem@special: 为短引用功能添加。	63
v1.6a – 1990/05/24		v1.7a – 1992/02/28	
\meta: 修正额外空格错误。	58	\DeleteShortVerb: 检查前一个匹 配的 \MakeShortVerb 以避免错 误。	61
v1.6b – 1990/06/15			
CodelineNo: 将 \rm 移到			

\wrong@table: 移至正确的类别码位置, 以便正常工作。	65	v1.7c – 1992/03/24	\@verbatim: 在\par 中添加了\interlinepenalty 来自 verbatim.sty	34
v1.7a – 1992/03/02			\macro@code: 向 \par 中添加 \interlinepenalty, 来自 verbatim.sty	27
\saved@macroname: 改变了用于更好排序的字符串。	55	v1.7c – 1992/03/25	\PercentIndex: 现在默认为修复了 bug 的 makeindex	50
v1.7a – 1992/03/04		v1.7c – 1992/03/26	\macro@font: 修改 OFSS 的字体更改。	29
theindex: 包含 multicols 的测试。	50		\mod@math@codes: 添加。	32
v1.7a – 1992/03/06			\OldMakeindex: 替换了 \NewMakeIndex	50
General: 添加了可派生出 docstrip 驱动文件的示例。	4	v1.7c – 1992/04/01	General: 从驱动文件中删除了 ltugboat.sty。	4
v1.7a – 1992/03/10		v1.7d – 1992/04/25	\Module: 使用无衬线字体表示模块。	32
\short@macro: 确存储储在 \macro@namepart 中的字符被视为“字母”, 以便索引排除功能正常工作。	39	v1.7f – 1992/05/16	\guard@level: 添加。	32
theglossary: 如果需要, 更改以在不使用 multicols 的情况下工作。	56		\slash@module: 考虑嵌套的守卫。	31
v1.7a – 1992/03/11		v1.7g – 1992/06/19	\special@escape@char: 使波浪号活动字符的操作移到定义之外	36
General: glo.ist 和 gind.ist 现在可由 doc.dtx 使用 docstrip 派生出来。	48	v1.7h – 1992/07/01	General: 在 gls 文件中关闭标题	53
关于 gind.ist 的使用说明。	11	v1.7i – 1992/07/11	\pm@module: 支持依赖于嵌套的字体。	31
添加了基本用法概要以明确阐述。	16		\slash@module: 添加计数器以确定何时切换到特殊字体。	31
v1.7a – 1992/03/12		StandardModuleDepth: 添加计数器。	32	
\ch@angle: 添加。	30	v1.7i – 1992/07/12	\@verbatim: 添加了\@@par 以清除可能的 \parshape。	34
\ch@percent: 添加。	30		verbatim*: 添加了 verbatim* 的变更定义。	34
\check@angle: 添加。	30			
\check@plus@etc: 添加。	30			
\CheckModules: 添加。	30			
\ifpm@module: 添加。	29			
\macro@font: 添加以支持模块区分。	29			
\Module: 添加。	32			
\pm@module: 添加。	31			
\slash@module: 添加。	31			
\theCodelineNo: 对于 NFSS, 使用 \reset@font。	67			
v1.7a – 1992/03/13				
\MacroFont: 为 NFSS 添加了 \reset@font	28			

v1.7i – 1992/07/17	\macro@cnt: 修复可能不再需要的 问题 35
\slash@module: 支持依赖于模块嵌 套的字体 31	v1.9n – 1994/04/28
v1.7j – 1992/08/14	\OnlyDescription: 如果未给出 \MaybeStop, 则忽略 \Finale 58
\codeline@wrindex: 添加了 \if@files。 66	v1.9o – 1994/05/08
v1.7m – 1992/10/11	\GetFileInfo: 添加宏 68
\macro@font: 将 sltt 作为默认字 体。 29	v1.9r – 1994/06/09
v1.8a – 1993/05/19	\maketitle: 添加了 \@makefnmark 和 \@makefntext 的新定义 . . . 59
\CodelineNumbered: 添加了宏 . . 66	v1.9t – 1995/05/11
v1.8b – 1993/09/21	General: 使用 \GetFileInfo 1
\@verbatim: 更改以符合新的 LaTeX verbatim, 处理更多连 字。 35	v1.9t – 1995/05/26
\macro@code: 更改以符合新的 LaTeX verbatim, 处理更多连 字。 27	\macro@font: 移除 \math@fontsfalse (不同的数 学设置 /pr1622) 29
v1.8c – 1993/10/25	\MacroFont: 移除 \math@fontsfalse (不同的数 学设置 /pr1622) 28
\macro@font: NFSS 标准 29	v1.9u – 1995/08/06
\MacroFont: NFSS 标准 28	\changes: 使用 \protected@edef 使用 \saved@macroname 的值来 查找外部层级的变更条目 55
\Module: NFSS 标准 32	\generalname: 添加宏 55
v1.9a – 1993/12/02	\saved@macroname: 现在默认为空 55
General: 升级至 LaTeX2e 1	v1.9v – 1995/11/03
v1.9b – 1993/12/03	\@MakeShortVerb: (DPC) 使用 \@shortvrbinf。 61
\macro@code: 强制从宏环境中的任 何标签。 26	\@shortvrbinf: (DPC) 添加宏 62
v1.9d – 1993/12/20	\DeleteShortVerb: (DPC) 使用 \@shortvrbinf。 62
General: 保护了 changes 条目。 . . . 1	v1.9w – 1995/12/27
v1.9e.2 – 1994/02/07	\AlsoImplementation: 添加宏 . . 57
\DeleteShortVerb: -js: 在 \DeleteShortVerb 中重置 'cc'⟨c⟩。 62	\index@prologue: 文本更改 51
\MakeShortVerb: -js: 检查 ⟨c⟩ 是否 已经是 \verb 的缩写。 60	v1.9w – 1995/12/29
v1.9h – 1994/02/10	\PrintChanges: 在使用后将该命令 变为 noop。 57
\PrintChanges: 使用 \@input@ 替 代 \@input。 57	\PrintIndex: 使用后将该命令转为 noop 53
\PrintIndex: 使用 \@input@ 替代 \@input 53	v1.9x – 1996/01/11
v1.9k – 1994/02/22	\index@prologue: 文本取决于所用 的代码行 51
\ch@angle: 将 < 设为活动字符 . . 30	

v1.9y – 1996/01/26	v2.1d – 2006/02/02
<code>\macro@font</code> : 支持兼容模式 29	General: 修正了 <code>\changes</code> 宏的描述。 14
<code>\MacroFont</code> : 支持兼容模式 28	
v1.9z – 1997/02/05	v2.1e – 2010/02/04
<code>\GetFileInfo</code> : 缺失的百分号	<code>\mod@math@codes</code> : 添加数学编码
<code>latex/2404</code> 68	(<code>pr/4096</code>) 32
v2.0a – 1998/05/16	v2.1f – 2016/02/12
<code>\macro@font</code> : 支持在导言部分更改	<code>\bslash@cnt</code> : 如果文件中未指定校
<code>\MacroFont</code> 29	验和, 则取消 <code>\Checksum</code> 检查 64
v2.0b – 1998/05/19	<code>\check@checksum</code> : 如果文件中未指
General: 重新在文档开头初始化私	定校验和, 则取消 <code>\Checksum</code>
有注释字符 (<code>pr2581</code>) 24	检查 63
v2.0e – 1998/12/28	v2.1g – 2016/02/15
<code>\short@macro</code> : 正确使用大小写转	<code>\changes</code> : 使用 <code>\saved@indexname</code> 55
换技巧。 39	<code>\GlossaryParms</code> : 默认使用类似左
v2.0i – 2000/05/21	对齐的设置 57
<code>\meta</code> : 新实现 (<code>pr/3170</code>) 58	<code>\saved@indexname</code> : 使用
v2.0j – 2000/05/22	<code>\saved@indexname</code> 55
<code>\index@prologue</code> : 措辞更清	v2.1h – 2018/02/01
晰? (<code>CAR pr/3202</code>) 51	<code>\DocstyleParms</code> : 只有在定义
v2.0k – 2000/05/26	了 <code>\DocStyleParms</code> 时才使用它
<code>\meta@font@select</code> : 添加宏	(以前的测试定义了它) 67
(<code>pr/3170</code>) 59	v2.1j – 2019/11/03
v2.0l – 2000/06/10	<code>verbatim*</code> : 内核现在设置
<code>\meta</code> : 修复变更 (<code>pr/3170</code>) 58	了 <code>\verbvisiblespace</code>
v2.0m – 2000/07/04	(<code>gh/205</code>) 34
<code>\meta</code> : 进一步修复变更 (<code>pr/3170</code>) 59	v2.1k – 2019/11/10
v2.0n – 2001/05/16	<code>verbatim*</code> : 将定义放入正确的命令
<code>\check@plus@etc</code> : 部分支持	中: <code>-(gh/205)</code> 34
<code>docstrip</code> 的“逐字”指令	v2.1l – 2019/12/16
(<code>pr/3331</code>) 31	<code>\MacroFont</code> : 对于扩展的 NFSS, 使
v2.1a – 2003/12/09	用 <code>\shapedefault</code> 而不是
<code>\MakeShortVerb*</code> : (HjG) 添加了 *	<code>\updefault</code> 28
形式 60	v2.1m – 2020/06/15
v2.1a – 2003/12/10	<code>\macro@code</code> : 垂直排版时清除
<code>\@shortvrbinf</code> : (HjG) 代表	<code>\@labels</code> (<code>gh/344</code>) 27
<code>\MakeShortVerb*</code> 添加第三个	v3.0a – 2018/03/04
参数 62	General: 与 <code>hypdoc</code> 包进行了接口
<code>\DeleteShortVerb</code> : (HjG) 如果不	对接 2
是短引用字符, 通知用户 62	集成了 <code>DoX</code> 包 2
	v3.0g – 2022/06/01
	<code>\changes</code> : 如果被要求, 展示变更日

期 (gh/531)	55	们知道该参数展开为一个单个字	
v3.0h – 2022/06/01		符串记号	45
General: 修复选择键名 (gh/750) .	25	\short@macro: 不再使用大小写转	
修复默认键名 (gh/750)	25	换技巧。	39
v3.0j – 2022/06/02		\SpecialShortIndex: 寻找正确的	
General: 使用 \providecommand 来		标记	46
定义 \pkg	1	v3.0m – 2022/11/13	
v3.0k – 2022/06/22		General: \verb 的重新定义已移除,	
General: 使用 \DeclareKeys	25	因为不再需要 (gh/953)	1
v3.0l – 2022/11/03		\@verbatim: 重新定义\@verbatim	
__doc_maybe_index_short:o: 我		以匹配核心定义 (gh/953)	34