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 12 日

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 TeX 代码提供代码文档的包。从那时起，它被广泛用于记录 LaTeX 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

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
\IndexInput

最后，⟨*special input commands*⟩ 部分应该包含一个或多个 `\DocInput{`⟨*file name*⟩`}` 和/或 `\IndexInput{`⟨*file name*⟩`}` 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 LaTeX 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]

```
1 ⟨*driver⟩
2 \documentclass{ltxdoc}
3
4 \usepackage[fontset=source]{ctex}
5 \usepackage[T1]{fontenc}
6 \usepackage{xspace}
7
8 \OnlyDescription
9
10 \EnableCrossrefs
11  %\DisableCrossrefs     % Say \DisableCrossrefs if index is ready
12 \CodelineIndex
13 \RecordChanges          % Gather update information
14 \SetupDoc{reportchangedates}
```

[1] 译者注：下面代码中，第 4 行在英文版中是不存在的，出于编译中文的需要，添加上了 ctex 宏包的引用，被 `macrocode` 环境抄录 (verbatim) 下来了；另外，第 19 行在英文原文中是 `\DocInput{doc.dtx}`，出于方便中文读者对照原文阅读以及后续增量更新的需求，在本项目中，`doc.dtx` 是英文版的源文件，而中文版译文的源文件则更名为 `doc-zh-cn.dtx`。

```
%\OnlyDescription      % comment out for implementation details
\setlength\hfuzz{15pt}  % don't show so many
\hbadness=7000          % over- and underfull box warnings
\begin{document}
   \DocInput{doc-zh-cn.dtx}
\end{document}
⟨/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 TEX 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 TEX 或 LATEX 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。[2] 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——TEX 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 TEX 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

[2] 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。

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。[3] 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*)
\MacroTopsep (*skip*)
\MacroIndent (*dimen*)
\MacroFont

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

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

[3] 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宏名称作为参数。

化由四个可用户自定义的宏完成。[4] 它们的名称分别是 \PrintDescribeMacro 和 \PrintDescribeEnv（定义了 \DescribeMacro 和 \DescribeEnv 的行为），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*⟩}

- 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。

[4] 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。

`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。[5]

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 `\|`。

[5] 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。

\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
\EnableCrossrefs

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[6]，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。[7]

\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。[8] 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。

\PageIndex
\CodelineIndex
\theCodelineNo

\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。

[6] 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。

[7] 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。

[8] 实际上，该行号是 macro 环境中第一个 macrocode 环境的第一行的行号。

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。[9] 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
\encapchar
\levelchar

\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
\SpecialMainEnvIndex

要为宏生成主索引条目，可以使用 \SpecialMainMacroIndex 宏[10]。它被称为“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。[11]

\SpecialMacroIndex
\SpecialEnvIndex

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

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。

v3 新增

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

[9] 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。

[10] 此宏由 macro 环境调用。

[11] 此宏由 environment 环境调用。

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
IndexColumns (*counter*) 数器 'IndexColumns' 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
\IndexPrologue 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
\IndexParms 局并允许下一篇文章在同一页上开始。索引列的格式（\columnssep 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：

\IndexMin	= 80.0pt	\MacroTopsep	= 7.0pt plus 2.0pt minus 2.0pt
\marginparwidth	= 126.0pt	\MacroIndent	= 9.94167pt
\marginparpush	= 0.0pt	\MacrocodeTopsep	= 3.0pt plus 1.2pt minus 1.0pt
\tolerance	= 1000		

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 ⟨*c*⟩ ——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{\⟨*c*⟩}，您随后可以使用 ⟨*c*⟩⟨*text*⟩⟨*c*⟩ 作为 \verb⟨*c*⟩⟨*text*⟩⟨*c*⟩ 的等价形式；类似地，*-形式的 \MakeShortVerb*{\⟨*c*⟩} 会给您 \verb*⟨*c*⟩⟨*text*⟩⟨*c*⟩ 的等价形式。如果您随后希望 ⟨*c*⟩ 恢复到其先前的含义，则使用 \DeleteShortVerb{\⟨*c*⟩}；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，$\mathcal{AMS}$-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 ⟨*dimen parameter*⟩ 的内容。

\OnlyDescription \MaybeStop \StopEventually 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。此宏[12] 有一个参数，在其中放置您希望在文档在此结束时打印的所有信息（例如通常在最后打印的参考文献）。如果缺少 \OnlyDescription 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。

v3 新增

\Finale

因此，您可以使用此功能为 TEX 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。

\maketitle \ps@titlepage

\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，LaTEX 2_ε 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}
```

[12] 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。

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。[13]

\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（LaTeX 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[14]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 \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 \PrintChanges 若要将变更信息写出，请在驱动文件中包含 \RecordChanges。要读取并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*) \GlossaryPrologue \GlossaryParms GlossaryColumns (*counter*) \GlossaryMin、\GlossaryPrologue 和 \GlossaryParms 宏以及计数器 GlossaryColumns 与 \Index...版本类似。（LaTeX 的“glossary”机制用于变更条目。）

\bslash 有时需要打印一个 \，但无法使用 \verb 命令，因为符号的 \catcode 已经被固定了。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

[13] 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。

[14] 在 2.6 版之前。

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 \CheckModules \Module \AltMacroFont docstrip 系统的“module”指令 [6] 通常被识别并调用特殊格式。可以在 .dtx 文件或驱动文件中使用 \CheckModules 和 \DontCheckModules 来开启或关闭此功能。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⟩
%␣␣␣␣\begin{macrocode}
⟨code for macro fred⟩
%␣␣␣␣\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 TEX 计数器（TEX counters）和不带反斜杠的 LATEX 计数器（LATEX counters），以及两种类型的 LATEX 长度寄存器（LATEX 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 \DescribeMaro 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 LaTeX 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 LaTeX 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 TEX 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 TEX 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
\CheckSum

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[15] 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，

[15] 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。

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 TEX 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 TEX 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 TEX 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（TEX），文档却被忽视了。在 TEX 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 系统是唯一的参考作品；相反，它是一个原型，足以在 TEX 世界中开发程序。它是足够的，但并非完全足够。[16] 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 TEX 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 LATEX 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。[17] 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 TEX 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 $\mathcal{AMS}$-TEX 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 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）；

[16] 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 TEX 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。

[17] 然而，这个论点是错误的，然而，它经常被引用。

'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. 19）；
- 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。

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 TEXbook). 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 LATEX.

[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 LATEX'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。

P

T