

CuS_{TE}X 宏集手册

Longaster

2024 年 4 月 21 日 v0.0.5b

CUSTEX

总目录

总目录	i	§ 3 ltx 模块	43
第一章 概述	1	§ 4 util 模块	43
第二章 文档接口	1	3.4.1 交叉引用、超链接和书签	43
§ 1 ltx 模块	2	3.4.2 向前查找和收集内容	46
2.1.1 参数处理器, Argument processors	5	3.4.3 分析记号	46
§ 2 util 模块	7	3.4.4 杂项	50
§ 3 页面布局, layout 模块	9	§ 5 box 模块	54
2.3.1 页面尺寸	9	3.5.1 为宽度固定和宽度可变的内容创 建超链接	54
2.3.2 主体尺寸	10	3.5.2 特殊的“水平”盒子	54
2.3.3 边距	13	§ 6 struct 模块	55
2.3.4 原有的变量	13	§ 7 L ^A T _E X 2 _ε 的 mark 机制	58
2.3.5 页眉页脚	14	第四章 章节标题和目录	59
2.3.6 杂项	15	§ 1 title class, 标题类	59
2.3.7 设置页眉页脚	15	§ 2 输出 L ^A T _E X 原始风格的目录	59
§ 4 盒子和填充, box 模块	17	§ 3 使用模板的目录	61
2.4.1 Framed	17	§ 4 etoc 风格的目录设置方式	61
2.4.2 Filler	19	§ 5 目录的内部处理方式	69
2.4.3 多栏文字	23	第五章 库的文档接口	69
2.4.4 额外增加文字的宽度	26	§ 1 pgf 库	69
2.4.5 旋转的盒子	26	5.1.1 文字渐变	70
§ 5 背景, bgfg 模块	27	5.1.2 在背景和前景中使用 Ti ^k Z 绘制	71
§ 6 索引, index 模块	28	§ 2 tcb 库	72
§ 7 文档结构, struct 模块	29	5.2.1 multicolumns/framed=tcbox..	72
2.7.1 初始化设置	30	§ 3 logo 库	72
2.7.2 编号	31	§ 4 doc 库	72
2.7.3 格式	32	§ 5 bnf 库	76
2.7.4 间距和缩进	32	§ 6 ref 库	79
2.7.5 浮动体	33	§ 7 box 库	80
2.7.6 杂项	33	5.7.1 paracol 环境	80
2.7.7 目录	35	5.7.2 multicolumns/framed=lfbox..	83
§ 8 buffer 模块	40	5.7.3 \fparbox _{P83} 和 \fvarbox _{P83} , 可设 置外框的命令	83
第三章 编程接口	40	§ 8 math 库	83
§ 1 L ^A T _E X 2 _ε 的钩子机制	41	§ 9 counter 库	83
§ 2 L ^A T _E X 2 _ε 的模板机制	42	§ 10 pdf 库	84

第六章 可单独加载的宏包	85	
§ 1 collectn	85	cus.module.util.tex 119
§ 2 lt3ekeys	90	cus.module.algo.tex 119
6.2.1 定义键	90	cus.module.layout.tex 119
6.2.2 设置键	90	cus.module.box.tex 120
6.2.3 lt3ekeys-elkernel	90	cus.module.bgfg.tex 120
6.2.4 定义命令——lt3ekeyscmd	90	cus.module.index.tex 120
6.2.5 定义命令扩展——lt3ekeysext	95	cus.module.struct.tex 120
TODO	102	cus.library.box.tex 121
索引	103	cus.library.math.tex 121
代码索引	103	cus.library.counter.tex 121
List of Hackings	119	cus.library.ref.tex 121
cus.module.ltx.tex	119	cus.library.pgf.tex 121
		cus.library.tcb.tex 122
		cus.library.pdf.tex 122
		lt3ekeys、lt3ekeyscmd 和 lt3ekeysext . . . 122
		lt3ekeys-elkernel 122
		lt3ekeys-collectn 122
		updatemarks 122

第一章 概述

目前 CusTeX 还处于早期的开发状态中，很多功能还并不完善。

CusTeX (CusLaTeX) 宏集意为 a Chinese User Scheme TeX (LaTeX), 为中文 LaTeX 用户定制的文档类框架。

对于排版外文文档, 已经有诸如 KOMA-Script、memoir 等优秀的文档类, 由于中文文档的特殊性, 直接使用它们虽然可能, 但这些文档类终究不是为中文用户设计的, 使用起来仍有些不便。而像 ctex 文档类, 则注重解决输出中文的最根本的问题, 要求它们具有像 KOMA-Script 文档类的完整功能不太可能。如此, 本宏集应运而生。

使用 CusTeX 可以方便地设置标题、目录、页面样式 (页面几何元素、页眉页脚等)、图表、背景、水印、边注、脚注、列表、索引、术语表等文档元素, 具有强大的可定制性。CusTeX 原生兼容 pgf 和 tcolorbox, 加载这两个宏包或使用 pgf 库可实现更多的功能 [TODO]。

CusTeX 通过模块 (module) 和库 (library) 来实现诸多功能。其中模块是核心部分, CusTeX 将自动加载它们; 库是提供额外功能的, 用户可以选择是否加载它们。库可能依赖其它模块和库, 但模块不会依赖库。

模块和库均可能加载其它宏包, 一般情况下, CusTeX 会自动加载这些模块并处理好它们的依赖和兼容性, 当用户需要加载其它宏包时, 最好通过 CusTeX 的宏包加载机制来加载它们 [TODO]。

CusTeX 支持 XeLaTeX、LuaLaTeX、upLaTeX、ApLaTeX (pLaTeX-ng) 等多种编译方式, 其中 LuaLaTeX、upLaTeX、ApLaTeX 还支持竖排 [TODO]。

CusTeX 还很好的支持和适配了通用驱动 (generic driver), 这是 LaTeX 2_ε 2022-06-01 中的新功能。

不兼容 beamer。

第二章 文档接口

CusTeX 定义的命令有的用于文档中, 有的则是面向开发者, 本章描述那些在文档中可能使用到的接口。

Logo. 输出 CusTeX, CusLaTeX。

```
\CusTeX
\CusLaTeX
```

\cussetup

```
\cussetup {⟨key-vals⟩}
\cussetup [⟨key path⟩] {⟨key-vals⟩}
\cussetup {
  ⟨key path1⟩ = {⟨key-vals1⟩} ,
  ⟨key path2⟩ = {⟨key-vals2⟩} ,
  ...
}
```

键值设置命令。

CuSTeX 的不同模块使用不同的 $\langle key path \rangle$ ，一般情况下，这些模块会提供自己的键值设置接口，为了使用 `\cussetup`_{P.2} 来设置这些键值，需要指定 $\langle key path \rangle$ 。

在本文档中，键的说明文字旁的表格中列出了键的完整写法， $\langle key path \rangle$ 即为灰色的部分。如键 `frame/outer-sep`_{P.18} 和 `frame/sep`_{P.18} 可以写成

```
\cussetup[frame]{outer-sep=0pt, sep=20pt}
或
\cussetup{ frame/outer-sep=0pt, frame/sep=20pt }
```

例 1\cussetstyle

```
\cussetstyle [⟨key path⟩] {⟨key⟩} {⟨key-vals⟩}
\cussetstyle * [⟨key path⟩] {⟨key⟩} {⟨code⟩}
```

自定义键。

带 * 的可使用一个参数，它代表键传入的值。

例如，若要为 `Framed`_{P.18} 新增键选项，则可以使用此命令。

```
\colorlet{frame color}{black}
\colorlet{fill color}{white}
\cussetstyle*[frame]{frame color}{\colorlet{frame color}{#1}}
\cussetstyle*[frame]{fill color}{\colorlet{fill color}{#1}}
\cussetstyle[frame]{color frame}{%
  frame={\setlength{\fboxsep}{#1\cusframesep}%
    \setlength{\fboxrule}{#1\cusframerule}%
    \fcolorbox{frame color}{fill color}}%
}
\begin{Framed}[frame color=blue, fill color=blue!20, color frame=2]
  \zhlipsum[1]
\end{Framed}
```

例 2

§ 1 ltx 模块

ltx 模块，本模块封装或提供一些 L^AT_EX 2_ε 的接口。

\csstring *

```
\csstring ⟨cs⟩
```

提取控制序列 $\langle cs \rangle$ 的名字。

```
\ttfamily \csstring\CusTeX \quad \csstring\l
.....
CusTeX |
```

例 3\Replicate *

```
\Replicate {⟨num expr⟩} {⟨code⟩}
```

重复 $\langle code \rangle$ $\langle num expr \rangle$ 次。

```
\lo  {\langle material \rangle}
\hi  {\langle material \rangle}
\lohi {\langle lo material \rangle} {\langle hi material \rangle}
```

```
\lo
\hi
\lohi
```

在数学模式中, 它们相当于 $\{\}_\{\dots\}$ 、 $\{\}^\{\dots\}$ 、 $\{\dots\}_\{\dots\}$, 在文本模式中, 它们也可直接使用, `\lo` 相当于 `\textsubscript`, `\hi` 相当于 `\textsuperscript`。

`\Large` 字_下^上, 字_下^上。\$ $H_{\text{lo}\{u\}\text{hi}\{n\}} H_{\text{lohi}\{u\}\{n\}}$ \$. 例 4

字_下^上, 字_下^上。 $H_u^n H_u^n$.

```
\makelapbox {\langle text \rangle}
\makelapbox [\langle width \rangle] [\langle pos \rangle] {\langle text \rangle}
\makelapbox [\langle width \rangle] [\langle pos \rangle] [\langle lap to \rangle] {\langle text \rangle}
\parlapbox {\langle width \rangle} {\langle text \rangle}
\parlapbox [\langle pos \rangle] [\langle height \rangle] [\langle inner-pos \rangle] {\langle width \rangle} {\langle text \rangle}
\parlapbox [\langle pos \rangle] [\langle height \rangle] [\langle inner-pos \rangle] [\langle lap to \rangle] {\langle width \rangle} {\langle text \rangle}
```

```
\makelapbox
\parlapbox
```

`\makelapbox` 的用法和 `\makebox` 一样, 但是会把它向 $\langle lap to \rangle$ 侧重叠。若给出了 $\langle width \rangle$ 或它不为空, 则先把 $\langle text \rangle$ 放在宽为 $\langle width \rangle$ 的盒子中。 $\langle pos \rangle$ 可选值为 l、c、r, 分别表示把 $\langle text \rangle$ 放在盒子的左边、中间、右边, 默认值为 c。 $\langle lap to \rangle$ 的可选值为 l、c、r, 分别表示把 $\langle text \rangle$ (如果有 $\langle width \rangle$, 则是把宽为 $\langle width \rangle$ 的盒子) 嵌入到左、右的文字中或一半嵌入到左边一半嵌入到右边, $\langle lap to \rangle$ 的默认值为 $\langle pos \rangle$ 的值。

`\parlapbox` 的用法和 `\parbox` 一样, 但是会把它向 $\langle lap to \rangle$ 侧重叠。 $\langle pos \rangle$ 指定基线的位置, 可选值为 t、c、b, 分别表示第一行文字的基线位置、文字的中间以及末行文字的基线。的默认值为 c。若给出 $\langle height \rangle$, 则把文字放在高度为 $\langle height \rangle$ 的盒子中, 根据 $\langle inner-pos \rangle$ 来决定文字的垂直位置, 可选值为 t、c、b。

这里 `\makelapbox{是}` 文字。
这里 `\makelapbox[] [l]{是}` 文字。
这里 `\makelapbox[] [r]{是}` 文字。

例 5

这里文字。这是文字。这里文字。

```
\numberfixedwidth {\langle width \rangle} {\langle filler \rangle} {\langle printer \rangle} {\langle number \rangle}
\numberzerofill {\langle width \rangle} {\langle number \rangle}
```

```
\numberfixedwidth ☆
\numerzerofill ☆
```

先将 $\langle printer \rangle$ 作用于 $\langle number \rangle$, 然后用 $\langle filler \rangle$ 向左或向右填充, 填充 $\langle filler \rangle$ 的次数为 $width - \text{len}(\text{printer}(\langle number \rangle))$ 。当 $\langle width \rangle$ 小于 0 时, 在右边填充, 否则在左边填充。 $\langle printer \rangle$ 必须是可展的。

`\numberzerofill` 用 0 填充数字。

```
\ExplSyntaxOn
\numberfixedwidth { 6 } { 0 } { \int_to_Hex:n } { 42 } \quad
\numberfixedwidth { 6 } { 0 } { \int_to_Hex:n } { `好 } \quad
\numberzerofill { 6 } { 12478 }
\ExplSyntaxOff
```

例 6

00002A 00597D 012478

`\zkern` 相当于 `\kern 0pt\relax`。

<code>enumlist</code> <code>enumlist*</code>	<pre>\begin{enumlist} [⟨default label⟩] {⟨left⟩} {⟨indent⟩} {⟨label sep⟩} {⟨right⟩} ... \end{enumlist}</pre>
---	--

相当于 `list` 环境。

`⟨default label⟩` 为列表的标签，默认为空；`⟨left⟩` 为左侧间距；`⟨indent⟩` 为每段首行的缩进；`⟨label sep⟩` 为标签与首行的间距；`⟨right⟩` 为右侧间距。

带星号的环境还会设置段落间距和每项的间距为 `0pt`。

`\cusemoji`
`\cusemojitotalratio`
`\cusemojilowerratio`

`\cusemoji {⟨pic filename⟩}`

插入一张图片，它的（总）高度为当前文字的高度的 `\cusemojitotalratioP4` 倍，并向下移动 `\cusemojilowerratioP4` 个文字的高度。`\cusemojitotalratioP4` 和 `\cusemojilowerratioP4` 必须为 0 或 a/b ，其中 a, b 为非零整数。`\cusemojitotalratioP4` 默认为 $8/9$ ，如果是负数则上下翻转，但宽度也会变成负值。`\cusemojilowerratioP4` 默认为 $1/7$ ，如果是负数则向上移动。

可以用于数学模式，会根据是否处于上下标而改变大小。

需要用户自行加载 `graphicx` 宏包。

```
\newcommand{\bdhj}{\cusemoji{bd-huaaji.png}}
滑稽 \bdhj ; $ \sin\bdhj = 2^\bdhj $
```

例 7

滑稽 🤡; $\sin \text{🤡} = 2^{\text{🤡}}$

`\IfGraphicsExists`

`\IfGraphicsExists {⟨graphics name⟩} {⟨true code⟩} {⟨false code⟩}`

判断图片文件是否存在。`\@curr@file` 展开为此文件名，若无此文件则为 `\relax`。

它会自动查找 `\setgraphicspathP4` 设置的路径，且可以自动补全文件扩展名。

需要用户自行加载 `graphicx` 宏包。

`\InputIfGraphicsExists`

`\InputIfGraphicsExists * [⟨key-val list⟩] {⟨file⟩}`

如果图片存在时使用图片，否则什么也不做。

它会自动查找 `\setgraphicspathP4` 设置的路径，且可以自动补全文件扩展名。

需要用户自行加载 `graphicx` 宏包。

`\setinputpath`
`\setgraphicspath`

```
\setinputpath      {⟨path clist⟩}
\setinputpath      + {⟨path clist⟩}
\setgraphicspath    {⟨path clist⟩}
\setgraphicspath    + {⟨path clist⟩}
```

设置导入文件或导入图片时需要查找的路径。`⟨path clist⟩` 使用逗号分隔，且需使用 `/` 作为目录分隔符。带 `+` 的命令为附加到原有的设置之后。

使用 `\setgraphicspathP4` 时需要自行加载 `graphicx` 宏包。

2.1.1 参数处理器，Argument processors

ltxcmd 提供了 \NewDocumentCommand 等命令来定义新的命令，每个参数可以使用“参数处理器”来先行处理，再传递给实际的代码（或其它参数处理器），并提供了 \ReverseBoolean、\SplitArgument、\SplitList、\TrimSpaces 等几个参数处理器。

本模块提供了更多的处理器。

```
\ReplaceArgumentIf {<test function>} {<true replacement>} {<false replacement>}
```

```
\ReplaceArgumentIf
```

<test function> 需要三个参数，分别为要测试的值、true 分支、false 分支。当测试为真时，把参数替换为 *<true replacement>*，否则，替换为 *<false replacement>*。

```
\newcommand\mytestfake[3]{\ifthenelse{\equal{#1}{fake}}{#2}{#3}}
%\usepackage{ifthen}
\DeclareDocumentCommand\whatnews
{ >{ \ReplaceArgumentIf{\mytestfake}{true}{#1} } m }
{#1 news}
\whatnews{fake}, \whatnews{some}.
```

例 8

true news, some news.

上述代码定义了一个命令 \whatnews，它检查第一个参数是否为 fake，如果是，则替换为 true。

```
\ReplaceArgumentIfEqual {<tl>} {<true replacement>} {<false replacement>}
\ReplaceArgumentIfStrEqual {<str>} {<true replacement>} {<false replacement>}
```

```
\ReplaceArgumentIfEqual
\ReplaceArgumentIfStrEqual
```

判断参数是否等于 *<tl>*（或 *<str>*），如果是则替换为 *<true replacement>*，否则，替换为 *<false replacement>*。

```
\DeclareDocumentCommand\foo
{ >{ \ReplaceArgumentIfEqual{s}{c}{#2} } m
  >{ \ReplaceArgumentIfEqual{j}{m}{#2} } m }
{[#1][#2]}
```

例 9

```
\foo {k}{j}
\foo {s}{o}
```

[j][m][c][o]

上述代码定义了一个命令 \foo，它判断第一个参数是否为 s，如果是，则替换为 c，否则替换为第二个参数的值（在使用它的参数处理器之前的值）。判断第二个参数是否为 j，如果是则替换为 m，否则不变。

```
\ReplaceArgumentIfMatch {<regex>} {<true replacement>} {<false replacement>}
```

```
\ReplaceArgumentIfMatch
```

判断此参数是否匹配正则表达式 *<regex>*，如果是，则替换为 *<true replacement>*，否则，替换为 *<false replacement>*。

\ExpandArgument

\ExpandArgument {<spec>}

类似于 `\ExpandArgsPS1`，先使用 <spec> 指定的展开方式展开这个参数，再传递给实际的代码（或其它参数处理器）。

目前有效的 <spec> 为 `coVvex` 和 `p` 之一。前几个和 `\ExpandArgsPS1` 的类似，`p` 类似于 `x`，但那些被保护的命令和未定义的命令以及数学公式中的命令不会被展开。

此外，还有几个特殊的 `spec`：

`sS` — 把参数转化为字符串，（使用 `\detokenize`）；

`sX` — 和 `p` 完全一样，（使用 `\text_expand:n`）；

`sF` — 类似于 `x`，但不可展开的记号和未定义的命令被移除了，（使用 `\text_purify:n`）；

`sP` — 类似于 `sX`，速度更快，但数学公式中的命令会被展开，未定义的命令也会出错。

例 10

```
\DeclareDocumentCommand \faa
{
  >\ExpandArgument{p} m }{#1}
\DeclareDocumentCommand \fee
{ >\ExpandArgument{sS} >\ExpandArgument{p} m }{#1}
\DeclareDocumentCommand \fii
{ >\ExpandArgument{sS} >\ExpandArgument{sP} m }{#1}
\DeclareDocumentCommand \foo
{ >\ExpandArgument{sS} >\ExpandArgument{sF} m }{#1}
\newcommand{\mytextit}[1]{\textit{#1}}
\faa{\textbf{bfseries} \mytextit{itshape} $ \mytextit{math rm } a+b=c $}\par
\ttfamily
\fee{\textbf{bfseries} \mytextit{itshape} $ \mytextit{math rm } a+b=c $}\par
\fii{\textbf{bfseries} \mytextit{itshape} $ \mytextit{math rm } a+b=c $}\par
\foo{\textbf{bfseries} \mytextit{itshape} $ \mytextit{math rm } a+b=c $}\par
```

```
bfseries itshape math rm a + b = c
\textbf {bfseries} \textit {itshape} $ \mytextit {math rm }
a+b=c $
\protect \textbf {bfseries} \protect \textit {itshape} $
\protect \textit {math rm } a+b=c $
bfseries itshape $ \mytextit {math rm } a+b=c $
```

例 11

```
\DeclareDocumentCommand\oof
{ >{ \ReplaceArgumentIfMatch{\A.\Z}{0#1}{#1} } >{ \ExpandArgument{e} } m
  >{ \ReplaceArgumentIfMatch{(.{2,}|~\lcr))}{c}{#2} } m }
{[#1][#2]}

\oof {1}{m}
\oof {10}{mn}
\oof {jk}{r}
\oof {{jk}}{r}
\DeclareDocumentCommand\mytext{}{ab}% 不能被展开
\oof {\mytext}{m}
\DeclareExpandableDocumentCommand\mytext{}{ab}% 可以被展开
\oof {\mytext}{m}
```

```
[01][c] [10][c] [jk][r] [jk][r] [0ab][c] [ab][c]
```

上述代码定义了一个命令 `\oof`，它的第一个参数先被完全展开（使用 `\ExpandArgument` [§6](#)），再传递给后一个参数处理器，这个参数处理器判断此参数是否是单个记号，如果是，则在其左侧加上 0，否则保持不变。

它的第二个参数使用正则表达式 `(\{2,\}|\^1cr)` 进行判断，如果匹配则替换为 `c`，否则保持不变。

```
\RegexReplaceArgument <{<regex>} <{<regex replacement>}>
\RegexReplaceArgument + <{<regex>} <{<regex replacement>}>
```

```
\RegexReplaceArgument
```

在参数中使用 `<regex>` 查找，并用 `<replacement>` 替换之。

带 `+` 的替换所有，不带 `+` 的替换一次。

```
\DeclareDocumentCommand\foo
{ >{ \RegexReplaceArgument
  {(\d{2,4})[\^1cr]}(\d{1,2})[\^1cr]}
  {1/\2/\3}
}
m }
{#1}
\foo{1920/02/09}
\foo{1920-02-09}
.....
1920/02/09 1920/02/09
```

例 12

TeXhackers note: 以上这些正则表达式的匹配和替换使用的是 L^AT_EX3 的 `l3regex` 库中的命令，如 `\regex_match:nnTF`、`\regex_replace_once:nnN`、`\regex_replace_all:nnN`，支持的正则表达式语法请参考 `interface3.pdf`。

```
special-dischyph = <normal|opacity|none>
dischyph-opacity = <{0--1 之间的数}>
```

```
typo/special-dischyph
typo/dischyph-opacity
```

`special-dischyph` 可以让 `\-`（和 `\@dischyph`）显示的文字具有透明度。`normal` 为默认的显示效果。`none` 移除显示的字符。

`dischyph-opacity` 设置透明度，但不会直接修改 `special-dischyph`。

使用 `\DocumentMetadata` 后效果更好。

§ 2 util 模块

```
\MapClist <{<comma list>} <{<tokens>}>
\MapList <{<list>} <{<tokens>}>
\MapInteger [<{<initial value>}] [<{<step>}] <{<final value>} <{<function>}>
```

```
\MapClist ☆
\MapList ☆
\MapInteger ☆
```

`\MapClist` [§7](#) 使用 `<tokens>` 迭代逗号分隔的列表 `<comma list>`，它将 `<tokens>` 置于列表项之前。

`\MapList` [§7](#) 使用 `<tokens>` 迭代记号列表 `<list>`，它将 `<tokens>` 置于列表项之前。

`\MapInteger` [§7](#) 从 `<initial value>` 到 `<final value>` 以 `<step>` 为步长，来迭代 `<function>`。

`\IterateClist`
`\IterateList`
`\IterateInteger`

`\IterateClist {<comma list>} {<inline code>}`
`\IterateList {<list>} {<inline code>}`
`\IterateInteger [{<initial value>}] [{<step>}] {<final value>} {<inline code>}`

`\IterateClist`^{例 13} 使用 `<inline code>` 迭代逗号分隔的列表 `<comma list>`, `<inline code>` 可带一个参数 #1, 它为当前迭代项。

`\IterateList`^{例 13} 使用 `<inline code>` 迭代记号列表 `<list>`, `<inline code>` 可带一个参数 #1, 它为当前迭代项。

`\IterateInteger`^{例 13} 从 `<initial value>` 到 `<final value>` 以 `<step>` 为步长, 来迭代 `<inline code>`, `<inline code>` 可以带一个参数 #1, 它为当前整数。

`$ \MapClist{1,2,3,n}{a_} $ \quad $ \IterateClist{1,2,3}{a_{#1}+} a_n $` **例 13**

$$a_1 a_2 a_3 a_n \quad a_1 + a_2 + a_3 + a_n$$

`\IterateThread`

`\IterateThread {<comma list1>} {<comma list2>} {<inline code>}`
`\IterateThread * {<comma list1>} {<comma list2>} {<inline code>}`
`\IterateThread [{<n>}] {<comma list1>} ... {<comma listn>} {<inline code>}`
`\IterateThread * [{<n>}] {<comma list1>} ... {<comma listn>} {<inline code>}`
`\IterateThread [{<n>}] {<comma list1>} ... {<comma listn>} {<middle>} {<inline code>}`
`\IterateThread [{<n>}] {<comma list1>} ... {<comma listn>} {<middle>} [{<last>}] {<inline code>}`
`\IterateThread * [{<n>}] {<comma list1>} ... {<comma listn>} {<middle>} [{<last>}] {<inline code>}`

使用 `<inline code>` 迭代这 n 个 `<comma list>`, `<inline code>` 可接受 $n + 1$ 个参数, 其中第一个参数为索引, 其后的参数分别为诸列表的当前迭代项。当某一个列表结束时迭代终止, 多余的项被移除。 n 的可选值为 1 – 7, 即最多可使用 7 个列表。

使用 `<middle>` 来分隔各项, 最后两项用 `<last>` 分隔, 默认与 `<middle>` 一致。如未给出, 则为空, 即不在两项之间插入其它符号。

带 * 的版本保留空项和每项前后的空格, 不带 * 的则不保留。

若某个 `<comma list>` 为单个记号, 则将其展开一次。这样, 可以使用一个宏保存列表项。

`$ \IterateThread{a+b,c+d,e+f}{A+B,C+D,E+F}{\dfrac{#2}{#3}\geq} 0 $ \par` **例 14**
`$ \IterateThread {a+b, ,e+f}{A+B,C+D, }{\dfrac{#2}{#3}\geq} 0 $ \par`
`$ \IterateThread *{a+b, ,e+f}{A+B,C+D, }{\dfrac{#2}{#3}\geq} 0 $ \par`

$$\frac{a+b}{A+B} \geq \frac{c+d}{C+D} \geq \frac{e+f}{E+F} \geq 0$$

$$\frac{a+b}{A+B} \geq \frac{e+f}{C+D} \geq 0$$

$$\frac{a+b}{A+B} \geq \frac{e+f}{C+D} \geq 0$$

`$ \IterateThread[2]{1,2,3,n}{n,n-1,n-2,1}[+][+\cdots+]{a_{#2}\cdot b^{\#3}} $` **例 15**
`%= $ a_1\cdot b^{n+a_2}\cdot b^{n-1}+a_3\cdot b^{n-2}+\cdots+a_n\cdot b^1 $`

$$a_1 \cdot b^n + a_2 \cdot b^{n-1} + a_3 \cdot b^{n-2} + \cdots + a_n \cdot b^1$$

```
\ucchar {⟨unicode slot⟩}
\ucchars {⟨unicode slots⟩}
```

```
\ucchar ☆
\ucchars ☆
```

展开为 ⟨*unicode slot*⟩ 对应的 Unicode 字符。⟨*unicode slots*⟩ 为空格分隔的 Unicode 代码点。

```
\ucchar{"5982}: %
\ucchars{"75 "74 "69 "6C "6A21 "5757}。
```

例 16

如: util 模块。

```
\Verbatimize {⟨balanced tokens⟩}
\Verbatimize * ⟨token⟩ ⟨tokens⟩ ⟨token⟩
```

```
\Verbatimize
```

以 verbatim 的形式输出 ⟨*balanced tokens*⟩ 或 ⟨*tokens*⟩。

带 * 的版本作用与 \verb 类似, 由一对 ⟨*token*⟩ 包裹, 也支持一对 { } 包裹。只是它仍然使用当前字体。不能作为一个命令的参数。

不带 * 的版本可以作为另一个命令的参数, 但如下几个字符必须使用转义的形式: # \$ % { } \ ^, 即, 使用 \# \\$ \% \{ \} \^ \^。

```
\IfPageOdd {⟨true⟩} {⟨false⟩}
```

```
\IfPageOdd
\IfAbsPageOdd
```

判断当前页码是否为奇数。`\IfAbsPageOdd` 仅在 shipout 时有效 (如在 shipout/foreground, shipout/background, shipout, shipout/after 钩子中)。

平常使用时并不一定准确, ref 库改进了这一点, 见第 5.6 节。

同上, 但可展。

```
\@ifpageodd ☆
\@ifabspageodd ☆
```

§ 3 页面布局, layout 模块

layout 提供页面布局的相关接口。

```
\setuplayout {⟨layout key-val⟩}
\setuplayout [⟨preset name⟩] {⟨layout key-val⟩}
\setuplayout * [⟨preset name⟩] {⟨layout key-val⟩}
```

```
\setuplayout
```

设置布局。

第一个用法为直接设置页面布局。第二个除了设置布局外, 还将这个布局保存下来, 可供后续重复使用。第三个则仅保存布局, 而不设置这个布局。

可以在文档中间改变布局, 纸张大小也可改变。

键值接口大都直接使用 geometry 宏包的接口。具体用法说明可参见其说明文档。如未作说明, 则与 geometry 宏包提供的接口用法相同。

2.3.1 页面尺寸

```
papername|paper = {⟨papername⟩}
```

设置纸张大小。⟨*papername*⟩ 为预定义的纸张名, 大小写无关。

```
layout/papername
layout/paper
```

```
papersize = {⟨宽⟩,⟨高⟩} 或 {⟨宽⟩:⟨高⟩} 或 {⟨长度⟩}
paperwidth = {⟨宽⟩}
paperheight = {⟨高⟩}
```

```
layout/papersize
layout/paperwidth
layout/paperheight
```

设置纸张大小。

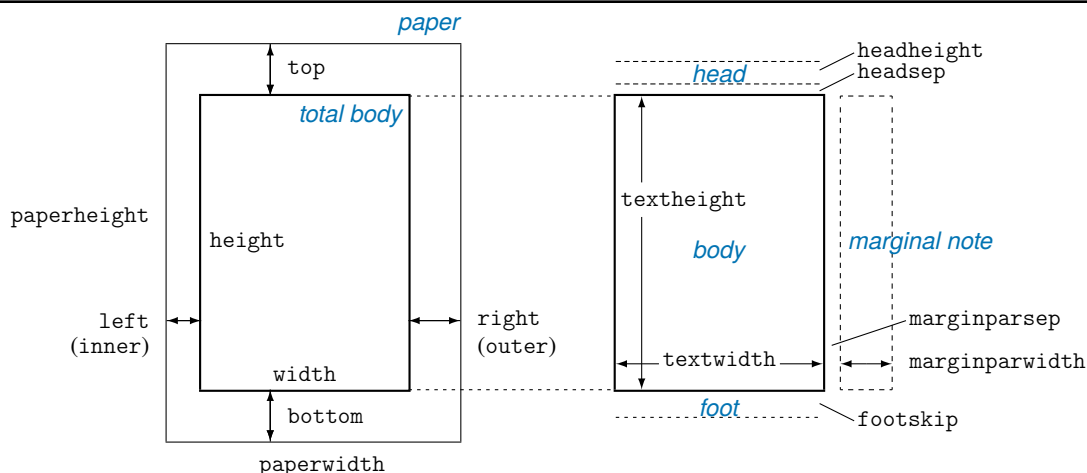


图 2.1: 长度变量

```
layout/paperorientation
layout/orientation
layout/landscape
layout/portrait
layout/direction
```

```
paperorientation|orientation = <landscape|portrait>
landscape
portrait
direction = <bigwidth|bigheight|normal|inverse>
```

不可设置值
不可设置值

设置纸张方向。使用 `portrait` 时, 纸张高度大于宽度。`landscape` 则反之。

`direction` 的 `bigheight` 和 `normal` 相当于 `portrait`, `bigwidth` 和 `inverse` 相当于 `landscape`。

使用 `papername` 等选项时, 将自动设置纸张方向, 使得实际纸张宽高与所给一致。

```
layout/layout
layout/layoutname
layout/layoutwidth
layout/layoutheight
layout/layoutsize
layout/layoutoffset
layout/layoutvoffset
layout/layoutoffset
layout/centerlayout
```

设置 *layout* 部分大小。

`layout` 或 `layoutname` 会根据纸张方向自动交换长宽, 因此纸张方向必须先于它们设置。

`centerlayout` 通过将 `layoutoffset` 和 `layoutvoffset` 设置为合适的值, 以将 *layout* 部分置于纸张中心。

见 `geometry` 宏包文档。

```
\layoutwidth
\layoutheight
\layoutloffset
\layouttoffset
\layoutroffset
\layoutboffset
```

layout 的大小, 以及距离纸张左、上、右、下侧的长度。在 `TikZ` 中使用时可能要用 `\the` 获取它们的值, 使用 `\dimeval`、`\dimexpr` 等命令以及 `calc` 宏包的功能时则不需要。

若是设置了 `twoside`, 它们不会自动切换左侧和右侧的值。

2.3.2 主体尺寸

此小节与 `geometry` 对应部分的用法和作用相同。

```
layout/hscale
layout/vscale
layout/scale
```

```
hscale = {(正实数)}
vscale = {(正实数)}
scale = {(hscale), (vscale)} 或 {(正实数)}
```

初始值: 0.7
初始值: 0.7

设置 *total part* 部分的宽高与纸张宽高的比率。

名称	宽 × 高	名称	宽 × 高	名称	宽 × 高
A0	841mm × 1189mm	B0	1000mm × 1414mm	C0	917mm × 1297mm
A1	594mm × 841mm	B1	707mm × 1000mm	C1	648mm × 917mm
A2	420mm × 594mm	B2	500mm × 707mm	C2	458mm × 648mm
A3	297mm × 420mm	B3	353mm × 500mm	C3	324mm × 458mm
A4	210mm × 297mm	B4	250mm × 353mm	C4	229mm × 324mm
A5	148mm × 210mm	B5	176mm × 250mm	C5	162mm × 229mm
A6	105mm × 148mm	B6	125mm × 176mm	C6	114mm × 162mm
b0j	1030mm × 1456mm	n0kai	787mm × 1092mm	b0kai	889mm × 1194mm
b1j	728mm × 1030mm	n2kai	787mm × 546mm	b2kai	889mm × 597mm
b2j	515mm × 728mm	n4kai	389mm × 546mm	b4kai	444mm × 597mm
b3j	364mm × 515mm	n6kai	370mm × 520mm	b8kai	420mm × 285mm
b4j	257mm × 364mm	n8kai	260mm × 370mm	b16kai	210mm × 285mm
b5j	182mm × 257mm	n16kai	185mm × 260mm	b32kai	142mm × 210mm
b6j	128mm × 182mm	n32kai	185mm × 130mm	ANSIA	8.5in × 11in
screen	225mm × 180mm	6kai	360mm × 390mm	ANSIB	11in × 17in
letter	8.5in × 11in	8kai	270mm × 390mm	ANSIC	17in × 22in
legal	8.5in × 14in	16kai	195mm × 270mm	ANSID	22in × 34in
executive	7.25in × 10.5in	32kai	195mm × 135mm	ANSIE	34in × 44in

表 2.1: 预定义的纸张名

```
totalwidth |width = {⟨长度⟩}
totalheight|height = {⟨长度⟩}
total = {⟨totalwidth⟩,⟨totalheight⟩} 或 {⟨长度⟩}
```

设置 *total part* 部分的宽高。

```
textwidth = {⟨长度⟩}
textheight = {⟨长度⟩}
body = {⟨textwidth⟩,⟨textheight⟩}
text = {⟨长度⟩}
```

设置 \textwidth、\textheight, 即 *body* 部分的宽高。

```
lines = {⟨行数⟩}
```

根据 ⟨行数⟩ 设置 textheight。⟨行数⟩ 一般为正整数。

```
includehead = ⟨true|false⟩
includefoot = ⟨true|false⟩
includeheadfoot|includehf = ⟨true|false⟩
```

控制是否将页眉 (\headheight、\headsep)、页脚 (\footskip) 计入 *total part* 部分中。

```
includemarginpar|includemp = ⟨true|false⟩
```

控制是否将旁注 (\marginparwidth、\marginparsep) 计入 *body* 部分中。

layout/totalwidth
layout/width
layout/totalheight
layout/height
layout/total

layout/textwidth
layout/textheight
layout/body
layout/text

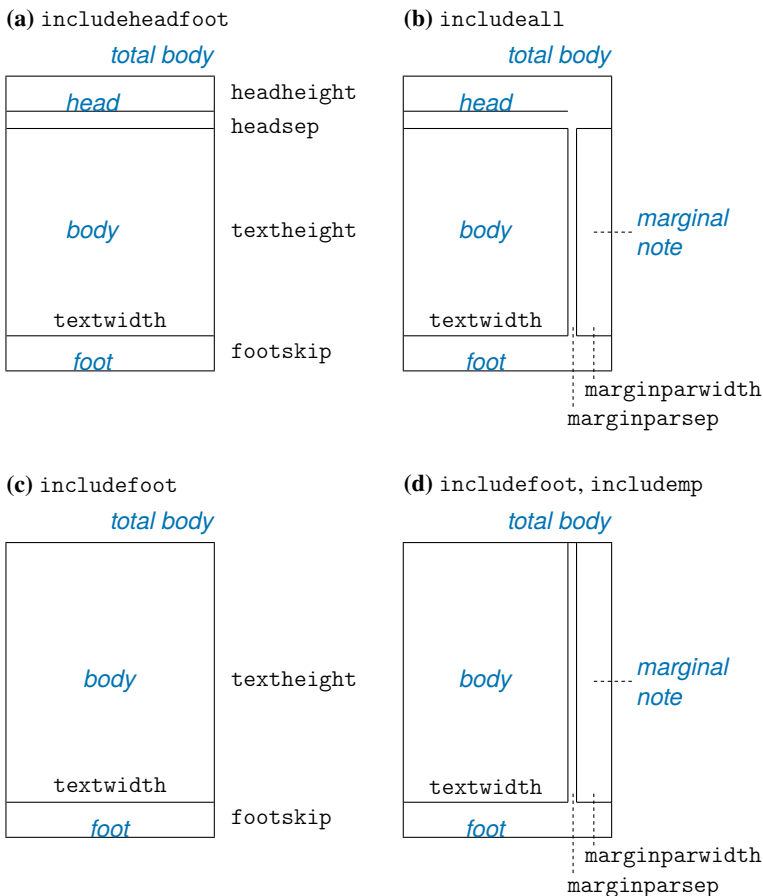
layout/lines

layout/includehead
layout/includefoot
layout/includeheadfoot
layout/includehf

layout/includemarginpar
layout/includemp

<div>layout/includeall</div>	<div>includeall = $\langle \text{true} \text{false} \rangle$</div> <div>初始值: false</div> <div>设置 includeheadfoot 及 includemarginpar。</div>
<div>layout/ignorehead</div> <div>layout/ignorefoot</div> <div>layout/ignoreheadfoot</div> <div>layout/ignorehf</div>	<div>ignorehead = $\langle \text{true} \text{false} \rangle$</div> <div>ignorefoot = $\langle \text{true} \text{false} \rangle$</div> <div>ignoreheadfoot ignorehf = $\langle \text{true} \text{false} \rangle$</div> <div>初始值: false</div> <div>初始值: false</div> <div>在计算垂直方向的尺寸时, 不考虑页眉、页脚。但不修改页眉页脚的尺寸。</div>
<div>layout/ignoremarginpar</div> <div>layout/ignoremp</div>	<div>ignoremarginpar ignoremp = $\langle \text{true} \text{false} \rangle$</div> <div>初始值: false</div> <div>在计算水平方向的尺寸时, 不考虑旁注的尺寸。但不修改旁注的尺寸。</div>
<div>layout/ignoreall</div>	<div>ignoreall = $\langle \text{true} \text{false} \rangle$</div> <div>初始值: false</div> <div>设置 ignoreheadfoot 及 ignoremarginpar。</div>
<div>layout/heightrounded</div>	<div>heightrounded = $\langle \text{true} \text{false} \rangle$</div> <div>初始值: false</div> <div>如果设置为真, 则将 textheight 设置为不小于原 textheight 且满足关系: $n \times \backslash \text{baselineskip} + \backslash \text{topskip}$的最小值。</div>

图 2.2
不同模式下的 *total body*。(a) includeheadfoot, (b) includeall, (c) includefoot 及 (d) includefoot, includemp。如果 reversemarginpar 设置为真, 则交换 *marginal note* 与 *body* 的位置。如果设置了 twoside, 则依据奇偶页交换 *marginal note*。



<div>layout/hdivide</div> <div>layout/vdivide</div> <div>layout/divide</div>	<div>hdivide = $\{ \langle \text{left margin} \rangle, \langle \text{width} \rangle, \langle \text{right margin} \rangle \}$</div> <div>vdivide = $\{ \langle \text{top margin} \rangle, \langle \text{height} \rangle, \langle \text{bottom margin} \rangle \}$</div> <div>divide = $\{ \langle \text{length}_1 \rangle, \langle \text{length}_2 \rangle, \langle \text{length}_3 \rangle \}$</div> <div>设置两个值, 将另一个留空或 *。</div>
--	--

`total body` 的边界距离 `layout` 边界的左、上、右、下侧的长度。在 `TikZ` 中使用时可能需要用 `\the` 获取它们的值, 使用 `\dimeval`、`\dimexpr` 等命令以及 `calc` 宏包的功能时则不需要。

若是设置了 `twoside`, 它们不会自动切换左侧和右侧的值。

2.3.3 边距

```
lmargin|leftmargin |left |inner = {⟨内侧边距⟩}
rmargin|rightmargin|right|outer = {⟨外侧边距⟩}
hmargin|horizontalmargin = {⟨inner⟩,⟨outer⟩} 或 {⟨水平边距⟩}
```

设置内外侧边距。注意, 不论是否使用 `twoside`, 它们的含义都是相同的。

```
tmargin|topmargin |top = {⟨顶部边距⟩}
bmargin|bottommargin|bottom = {⟨底部边距⟩}
vmargin|verticalmargin = {⟨top⟩,⟨bottom⟩} 或 {⟨垂直边距⟩}
```

设置上下边距。

```
hmarginratio|horizontalmarginratio = {⟨inner ratio⟩}:{⟨outer ratio⟩}
vmarginratio|verticalmarginratio = {⟨top ratio⟩}:{⟨bottom ratio⟩} 初始值: 2:3
marginratio = {⟨hmargin ratio⟩,⟨vmargin ratio⟩} 或 {⟨margin ratio⟩}
```

设置内外边距、上下边距的比率。

使用 `oneside` 时 `hmarginratio` 初始为 1:1, 使用 `twoside` 时 `hmarginratio` 初始为 2:3。

```
hcentering = ⟨true|false⟩
vcentering = ⟨true|false⟩ 初始值: false
centering = ⟨true|false⟩
```

设置 `hmarginratio`、`vmarginratio` 为 1:1。

```
twoside 不可设置值
asymmetric 不可设置值
reversemarginpar|reversemp = ⟨true|false⟩ 初始值: false
```

设置左右边距根据奇偶页进行切换。 `asymmetric` 并不实际切换, 而是修改长度, 见 `geometry` 宏包文档。

```
bindingoffset = {⟨长度⟩}
```

从内侧移除 `⟨长度⟩`。

2.3.4 原有的变量

本小节描述几个 `LATEX 2ε` 原有的长度变量。

```
footnotesep = {⟨弹性长度⟩}
```

设置 `\skip\footins`, 即正文底部与脚注顶部的距离。

```
\bodylmargin
\bodytmargin
\bodyrmargin
\bodybmargin
```

```
layout/leftmargin
layout/left
layout/lmargin
layout/inner
layout/rightmargin
layout/right
layout/rmargin
layout/outer
layout/hmargin
layout/horizontalmargin
```

```
layout/topmargin
layout/top
layout/tmargin
layout/bottommargin
layout/bottom
layout/bmargin
layout/verticalmargin
```

```
layout/horizontalmarginratio
layout/hmarginratio
layout/verticalmarginratio
layout/vmarginratio
layout/marginratio
```

```
layout/hcentering
layout/vcentering
layout/centering
```

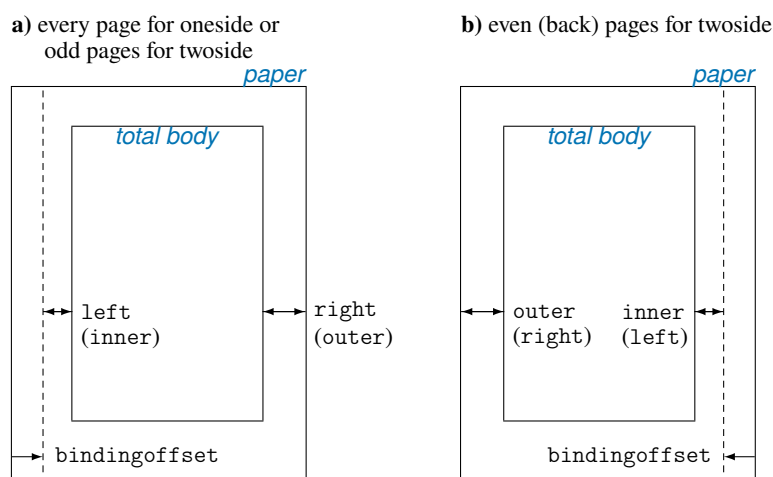
```
layout/twoside
layout/asymmetric
layout/reversemarginpar
layout/reversemp
```

```
layout/bindingoffset
```

```
layout/footnotesep
```

图 2.3

The option `bindingoffset` adds the specified length to the inner margin. Note that `twoside` option swaps the horizontal margins and the marginal notes together with `bindingoffset` on even pages (see **b**), but `asymmetric` option suppresses the swap of the margins and marginal notes (but `bindingoffset` is still swapped).



`layout/marginparwidth`
`layout/marginpar`
`layout/marginparsep`
`layout/nomarginpar`
`layout/nomp`

`marginparwidth|marginpar` = `{<长度>}`
`marginparsep` = `{<长度>}`
`nomarginpar|nomp`

不可设置值

设置旁注宽度及旁注与正文的距离。`nomarginpar` 将它们设置为 0pt。

`layout/columnsep`
`layout/twocolumn`
`layout/onecolumn`

`columnsep` = `{<长度>}`
`twocolumn`
`onecolumn`

不可设置值

不可设置值

设置 `\columnsep`，即两栏之间的距离。

`layout/hoffset`
`layout/voffset`
`layout/offset`

`hoffset` = `{<长度>}`
`voffset` = `{<长度>}`
`offset` = `{<hoffset>,<voffset>}` 或 `{<长度>}`

设置 `\hoffset`、`\voffset`。

2.3.5 页眉页脚

`layout/headheight`
`layout/head`
`layout/headsep`

`head|headheight` = `{<长度>}`
`headsep` = `{<长度>}`
`nohead`

`headheight` 设置 `\headheight`，即页眉的高度。

`headsep` 设置 `\headsep`，即页眉与正文之间的距离。

`nohead` 将它们设置为 0pt。

`layout/footskip`
`layout/foot`
`layout/nofoot`

`footskip|foot` = `{<弹性长度>}`

设置 `\footskip`，即正文最后一行的基线与页脚基线的距离。

`nofoot` 将它设置为 0pt。

`layout/noheadfoot`
`layout/nohf`

`noheadfoot|nohf`

不可设置值

同时设置 `nohead` 和 `nofoot`

```
headoffset = {⟨长度⟩}
```

初始值: 0pt

```
headoffset = [⟨位置⟩] {⟨长度⟩}
```

```
layout/headoffset
layout/footoffset
layout/hfoffset
```

设置页眉页脚偏移量。

⟨位置⟩为 O、E 与 L、C、R 的组合。这五个值分别代表奇偶、左中右。不区分大小写。

若 ⟨长度⟩为正值, 则相较于 `textwidth` 伸长 ⟨长度⟩。否则, 缩短 ⟨长度⟩。

此选项在直排文档中可能无效。

2.3.6 杂项

本小节列出其它几个选项。未列出的选项请参考 `geometry` 宏包文档。

```
showframe = ⟨true|false⟩
```

初始值: false

```
showcrop = ⟨true|false⟩
```

初始值: false

```
showmarking|marking = ⟨true|false⟩
```

初始值: false

```
layout/showframe
layout/showcrop
layout/showmarking
layout/marking
```

`showframe` 显示各部分的外框。`showcrop` 在 `layout` 四角显示裁剪标记。`marking` 在各部分着以彩色背景。

```
preset|name = {⟨preset name⟩}
```

```
layout/preset
layout/name
```

使用预设值 ⟨`preset name`⟩。

2.3.7 设置页眉页脚

本小节设置页眉页脚内容的接口。关于设置页眉页脚位置和高度的接口, 见第 2.3.5 小节。

本节所述内容可能在直排文档中不可用。

本节所述的功能主要通过 `fancyhdr` 实现。

```
\usepagestyle {⟨pagestyle⟩}
```

```
\usepagestyle
\usethispagestyle
```

`\usepagestyleℓ15` 页眉页脚的样式 ⟨`pagestyle`⟩。`\usethispagestyleℓ15` 设置本页的样式。

有一个预定义的样式 `totalempy`, 它将页眉页脚设置为空, 并将页眉页脚横线的厚度设为 0pt。

只读。展开为当前使用的样式名 ⟨`pagestyle`⟩ 或 `\usethispagestyleℓ15` 设置的样式名。

```
\@currpagestyle *
\@specialstyle *
```

```
\setpagestyle {⟨pagestyle⟩} {⟨code⟩}
```

```
\setpagestyle {⟨pagestyle1⟩} [⟨pagestyle2⟩] {⟨code⟩}
```

```
\setpagestyle * {⟨pagestyle1⟩} [⟨pagestyle2⟩] {⟨code⟩}
```

```
\setpagestyle . {⟨pagestyle1⟩} [⟨pagestyle2⟩] {⟨code⟩}
```

```
\setpagestyle + {⟨pagestyle1⟩} [⟨pagestyle2⟩] {⟨code⟩}
```

```
\coppagestyle {⟨pagestyle1⟩} {⟨pagestyle2⟩}
```

```
\setpagestyle
\coppagestyle
```

设置样式 {⟨`pagestyle`⟩}, 或基于样式 ⟨`pagestyle2⟩`⟩ 设置 ⟨`pagestyle1⟩`⟩。如果 ⟨`pagestyle2⟩`⟩ 发生改变, 那么 ⟨`pagestyle1⟩`⟩ 也可能随之改变。

不带其它符号的, 仅设置而不使用。带 * 的, 还会立刻使用该样式, 但不会修改 `\usethispagestyleℓ15` 所设置的样式。带 . 的, 修改 `\usethispagestyleℓ15` 所设置的样式。带 + 的, 相当于同时使用 `\usepagestyleℓ15` 和 `\usethispagestyleℓ15`。

`\coppagestyleℓ15` 把样式 ⟨`pagestyle2⟩`⟩ 复制给 ⟨`pagestyle1⟩`⟩。

<hr/> <code>\sethead</code> <code>\setfoot</code> <code>\setheadfoot</code> <code>\setleftthead</code> <code>\setcenterthead</code> <code>\setrightthead</code> <code>\setlefttfoot</code> <code>\setcentertfoot</code> <code>\setrighttfoot</code> <hr/>	<div> <code>\sethead</code> <code>{<code>}</code> <code>\sethead</code> [<code><位置></code>] <code>{<code>}</code> <code>\setcenterthead</code> <code>{<奇偶页>}</code> <code>\setcenterthead</code> [<code><偶数页></code>] <code>{<奇数页>}</code> </div> <p>设置页眉页脚的内容。</p> <p><code><位置></code> 为 O、E、L、C、R、H、F 此三类的组合。这七个值分别代表奇偶、左中右、页眉页脚。不区分大小写。</p> <p>如某一类未给出, 则视为该类的全部值都给出。但 <code>\sethead</code>_{P.16}、<code>\setfoot</code>_{P.16} 分别为 H、F。</p> <p>例如, 在 <code>\sethead</code>_{P.16} 中, L 代表 OLH, ELH。</p> <p>它们可以直接用在导言区和正文中, 将修改本页及其后面页面的页眉页脚。但最好用于 <code>\setpagestyle</code>_{P.15} 命令中, 统一设置页眉页脚。</p>
<hr/> <code>\setheadrulewidth</code> <code>\setfootrulewidth</code> <hr/>	<div> <code>\setheadrulewidth</code> <code>{<长度表达式>}</code> </div> <p>设置页眉、页脚横线的厚度。</p> <p>(即宏 <code>\headrulewidth</code>、<code>\footrulewidth</code> 的值。)</p>
<hr/> <code>\setheadruleskip</code> <code>\setfootruleskip</code> <hr/>	<div> <code>\setheadruleskip</code> <code>{<skip expr>}</code> </div> <p>设置页眉、页脚横线与页眉、页脚文字的距离。</p> <p>(即宏 <code>\headruleskip</code>、<code>\footruleskip</code> 的值。)</p>
<hr/> <code>\setheadrule</code> <code>\setfootrule</code> <hr/>	<div> <code>\setheadrule</code> <code>{<code>}</code> </div> <p>设置页眉、页脚的横线。页眉的横线的总高度最好为 0。</p>
<hr/> <code>\setheadinit</code> <code>\setfootinit</code> <code>\setheadfootinit</code> <hr/>	<div> <code>\setheadinit</code> <code>{<code>}</code> </div> <p>在输出页眉页脚前要执行的 <code><code></code>。</p>
<hr/> <code>\fancycenter</code> <hr/>	<div> <code>\fancycenter</code> <code>{<left>}</code> <code>{<center>}</code> <code>{<right>}</code> <code>\fancycenter</code> [<code><distance></code>] [<code><stretch></code>] <code>{<left>}</code> <code>{<center>}</code> <code>{<right>}</code> </div> <p>它创建一个盒子, 使得 <code><center></code> 位于当前行 (或盒子) 的中心。可以用于正文中。</p> <p><code><center></code> 的中心与 <code><left></code>、<code><right></code> 的中心的距离可能并不一致。</p> <div> <div> <code>\fancycenter{L}{CCCC}{RRRRRRRRRRRRRRRR}</code> </div> <div>例 17</div> </div> <hr/> <div> <div>L</div> <div>CCCC</div> <div>RRRRRRRRRRRRRRRR</div> </div>
<hr/> <code>\iftopfloat</code> <code>*</code> <code>\ifbotfloat</code> <code>*</code> <code>\iffloatpage</code> <code>*</code> <code>\iffootnote</code> <code>*</code> <hr/>	<div> <code>\iftopfloat</code> <code>{<true>}</code> <code>{<false>}</code> </div> <p>检测当前页是否顶部、底部有浮动体, 或当前页是否是浮动体页, 或当前页是否有脚注。</p>

§ 4 盒子和填充, box 模块

box 用于提供盒子构造、内容填充等内容。

```
\spreadtext <{width}> <{material}>
\spreadtext * <{width}> <{material}>
```

`\spreadtext`

在宽 $\langle width \rangle$ 的盒子中把 $\langle material \rangle$ 分散排列。

带星号的命令还会在 $\langle material \rangle$ 两端插入等量的间距。

此命令在 Cus \LaTeX 支持的引擎中都有效。在 Lua \LaTeX 引擎中, 需加载 `ctex` 或 `luatexja` 宏包; 在 Xe \LaTeX 引擎中, 需加载 `ctex` 或 `xeCJK` 宏包; 在 up \LaTeX 引擎中, 无需加载其它宏包。

```
\fbox{\spreadtext {5cm}{你好 Hello World 世界}}
\fbox{\spreadtext*{5cm}{你好 Hello World 世界}}
```

例 18

你 好 Hello World 世 界
你 好 Hello World 世 界

```
\useinfinitespace <{fil dimen}>
```

`\useinfinitespace`

把单词或 CJK 文字之间的距离设置为无限可伸长的间距, 它比 `\spreadtext`₁₇ 更为底层。

$\langle fil\ dimen \rangle$ 为 `factorfil` 或 `factorfill` 或 `factorfilll`。 *factor* 为一个实数。对于 `fil`、`fill`、`filll`, 它们的阶数依次增加, 在同一个盒子中, 高阶优先生效, 低阶的无效。若阶数相同, 则间距的长度由 *factor* 控制, 在同一个盒子中相同的 *factor* 保证间距相等。

它应当在一个组中使用。

```
\parbox{8\ccwd}{\useinfinitespace{1fil}%
\rightskip0pt plus 1fil \parfillskip-\rightskip
测试一下哦哈哈 ac b 哦哈}
```

例 19

测试一下哦哈哈
ac b 哦 哈

```
\fbox{\makebox[10cm][s]{%
{\useinfinitespace{1fil}梅川库子}%
\hspace{0pt plus 3fil}%
{\useinfinitespace{1.5fil}先生}%
}}
```

例 20

梅 川 库 子 先 生

2.4.1 Framed

box 模块定义了一个简易的可跨页的盒子环境 `Framed`, 相较于 `tcolorbox` 宏包提供的环境来说, 使用此环境的速度更快。它也可配合 `tcolorbox` 宏包使用。

<code>Framed</code>	<pre>\begin{Framed} [<i><frame key-val></i>] ... \end{Framed}</pre>	
	创建一个可跨页的盒子。若在另一个盒子内则不可跨页。	
<code>frame/outer-sep</code>	<code>outer-sep = {<skip expr>}</code>	初始值: <code>8pt plus 8pt minus 6pt</code>
	设置盒子与上下文的间距。	
<code>frame/sep</code>	<code>sep = {<长度表达式>}</code>	初始值: <code>3\fbboxsep</code>
	设置变量 <code>\cusframesep</code> , 即盒子外框与内容的间距。	
<code>frame/rule-width</code>	<code>rule-width = {<长度表达式>}</code>	初始值: <code>\fbboxrule</code>
	设置变量 <code>\cusframerule</code> , 即盒子外框的厚度。	
<code>frame/frame</code> <code>frame/frame*</code> <code>frame/first</code> <code>frame/first*</code> <code>frame/middle</code> <code>frame/middle*</code> <code>frame/last</code> <code>frame/last*</code> <code>frame/whole</code> <code>frame/whole*</code>	<pre>frame = {<code>} frame* = {<code width 1 parameter>}</pre>	
	<code>frame</code> 设置盒子外框。 <code>first</code> , <code>middle</code> , <code>last</code> 设置分页盒子的前、中、后三部分的外框。 <code>whole</code> 设置未分页盒子的外框。 <code><code></code> 其后可接一个参数, 这个参数为分页后的盒子。 <code><code with 1 parameter></code> 显式给出变量 #1。	
<code>frame/init</code>	<code>init = {<code>}</code>	
	盒子中执行的初始化代码。	
<code>frame/width</code> <code>frame/ratio</code>	<code>width = {<长度表达式>}</code> <code>ratio = {<数值表达式>}</code>	初始值: <code>\textwidth</code> 初始值: <code>1</code>
	<code>ratio</code> 设置盒子内容 (含边框) 占 <code>width</code> 的比率。	
<code>frame/align</code> <code>frame/left</code> <code>frame/center</code> <code>frame/right</code> <code>frame/inner</code> <code>frame/outer</code>	<code>align = <left center right inner outer></code>	初始值: <code>center</code>
	设置水平对齐方式。当 $0 < \langle ratio \rangle < 1$ 时才有效。	

```
\begin{Framed}[ratio=.8,center,  
rule-width=2pt,  
frame={\setlength{\fbboxsep}{\cusframesep}%  
       \setlength{\fbboxrule}{\cusframerule}%  
       \fcolorbox{purple}{cyan!50}}]  
\zhlipsum[9][name=zhufu]  
\end{Framed}
```

例 21

我很悚然，一见她的眼钉着我的，背上也就遭了芒刺一般，比在学校里遇到不及豫防的临时考，教师又偏是站在身旁的时候，惶急得多了。对于魂灵的有无，我自己是向来毫不介意的；但在此刻，怎样回答她好呢？我在极短期的踌躇中，想，这里的人照例相信鬼，然而她，却疑惑了，——或者不如说希望：希望其有，又希望其无……，人何必增添末路的人的苦恼，一为她起见，不如说有罢。

ignore-warnings

不可设置值

frame/ignore-warnings

忽略部分警告。

2.4.2 Filler

“filler” 是用以填充空间的那部分内容。如 \LaTeX 2_ϵ 的 `\hrulefill` 是用水平直线填充，`\dotfill` 是用句点填充，`\hspace` 是用空白填充。

`box` 提供了几个创建 filler 的命令。

```
\dashfiller
\dashfiller {<filler width>}
\dashfiller [<raise>] [<sep width>] [<rule width>]
\dashfiller [<raise>] {<filler width>} [<sep width>] [<rule width>]
```

`\dashfiller`

使用虚线填充，虚线宽和虚线间的距离近似为 `<sep width>`，使得虚线充满 `<filler width>`。

- `<filler width>` 为总宽度，默认值为 `\linewidth`。
- `<raise>` 为虚线升高的高度，默认为 `0pt`。
- `<sep width>` 为虚线宽和虚线间的距离，默认为 `1ex`。
- `<rule width>` 为虚线的厚度，默认为 `0.4pt`。

```
\noindent\llap{\}\dashfiller \par % 总长为 \linewidth
\noindent\llap{\}\dashfiller [.5ex] \par % 升高 .5ex
% 升高 .5ex, 宽 3pt, 注意第二个可选参数前不能有空格
\noindent\llap{\}\dashfiller [.5ex][3pt] \par
```

例 22

```
|-----|
|-----|
|-----|
```

`\filler [<filler key-val>]``\filler`

使用给定内容填充。

```
size = {<skip expr>}
size* = {<长度>}
```

filler/size

filler/size*

设置填充的长度。`size*` 填充的长度是弹性的。

注意在行间数学模式中 (`equation`、`align` 等环境) 弹性的那部分长度无效。

```
filler/space
filler/hspace
filler/hspace*
filler/vspace
filler/vspace*
filler/not-space
```

```
space    = <{*|<code>>
hspace   = <{*|<skip expr>>
hspace*  = <{*|<skip expr>>
vspace   = <{*|<skip expr>>
vspace*  = <{*|<skip expr>>
not-space
```

不可设置值

使用空白填充。使用它后, 其它选项无效。

`<code>` 是填充的内容, 如 `\hspace{1cm}`, `\vspace*{1cm}`。

`hspace` 相当于设置 `space=\hspace{<skip expr>}`。

`hspace*` 相当于设置 `space=\breakablehspace{<skip expr>}`。

`vspace` 相当于设置 `space=\vspace{<skip expr>}`。

`vspace*` 相当于设置 `space=\breakablevspace{<skip expr>}`。

当值为 `*` 时, 使用 `filler/sizeℰ19` 或 `filler/size*ℰ19` 设置的值。若设置 `space=*`, 则会根据是否处于垂直模式而使用 `\vspace` 或 `\hspace`。

由于用 `space` 填充的优先级最高, 若设置使用 `space` 填充后, 要使用其它类型来填充, 需使用 `not-space` 或将 `space` 设置为空。若后续仍设置了 `space`, 则仍会使用 `space` 填充。

`\breakablehspace` 和 `\breakablevspace` 是可断开的水平或垂直空白。实际给出的空白与要求的有一定的误差。

左 `\fbox{\strut \filler [hspace=5cm]}` 右间隔约 5cm。

左 `\filler[space]` 右拉开。

左 `\filler[space]` 中 `\filler[space]` 右拉开。

左
左
左

右间隔约 5cm。

右拉开。

中

右拉开。

例 23

```
filler/color
```

```
color = {<color expr>}
```

设置颜色 `cusfiller`, 即填充的颜色。

```
filler/content
filler/box
filler/box*
filler/clear-box
```

```
content = {<content>}
box      = {<content>}
box*     = {<长度表达式>} {<content>}
clear-content
clear-box
not-content
```

不可设置值
不可设置值
不可设置值

使用长 `<长度表达式>` 的 `<content>` 填充。

`content` 和 `box` 选项基本一致, 只是 `content` 会自动设置颜色, 而 `box` 则需使用 `\color` 或 `\color_select:n` 来设置颜色。

使用 `content` 将使用 `<content>` 的自然宽度, 而 `box*` 则使用指定的宽度。

当设置了 `box` 或 `box*` 后, `content` 无效, 除非使用 `clear-box` 清除 `box`。

第 20 页

<code>dash sep = {⟨dash length⟩}</code>	初始值: 0pt	<code>filler/dash</code>
<code>rule = {⟨rule width⟩}</code>	初始值: 0.4pt	<code>filler/sep</code>
<code>raise = {⟨raise height⟩}</code>	初始值: 0pt	<code>filler/rule</code>
<code>full = {true false}</code>	初始值: false	<code>filler/raise</code>
<code>is-dash</code>	不可设置值	<code>filler/full</code>
		<code>filler/is-dash</code>

使用虚线填充。

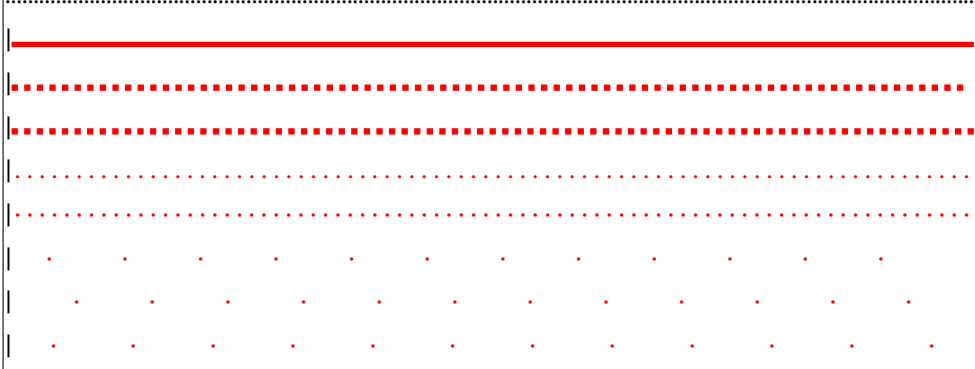
虚线宽和虚线间距为 `⟨dash length⟩`, 厚度为 `⟨rule width⟩`, 升高 `⟨raise height⟩`。

若 `⟨dash length⟩` 为 0pt, 则使用实线填充。

如果设置 `full` 为真, 则相当于 `\dashfiller`^{P19}。

<code>solid</code>	不可设置值	<code>filler/solid</code>
<code>dashed = {⟨长度⟩}</code>		<code>filler/dashed</code>
<code>dotted = {⟨间距⟩}</code>		<code>filler/dotted</code>
<code>cdotted = {⟨间距⟩}</code>		<code>filler/cdotted</code>

使用实线、虚线、句点或 `\cdot` 填充。

<pre> \def\BL{\noindent\llap{}% \BL \filler[color=red, solid, rule=2pt] \par \BL \filler[color=red, dashed, rule=0.5ex] \par \BL \filler[color=red, dashed, rule=0.5ex, full] \par \BL \filler[color=red, dotted] \par \BL \filler[color=red, cdotted] \par \BL \filler[color=red, cdotted=1cm, align] \par % 每个点都是对齐的 \BL \filler[color=red, cdotted=1cm, center] \par % 每个点的间距都是 1cm \BL \filler[color=red, cdotted=1cm, spread] % 每个点的间距都相等, 可能超过1cm </pre>	例 24
	

<code>type = {align center spread}</code>	初始值: align	<code>filler/type</code>
<code>align</code>	不可设置值	<code>filler/align</code>
<code>center</code>	不可设置值	<code>filler/center</code>
<code>spread</code>	不可设置值	<code>filler/spread</code>

构造填充的方式。

- `align`: 每个同种 `filler` 都是无限长的对齐的填充中的一部分, 因此, 它们处处都是对齐的;
- `center`: 把用以填充的盒子紧挨着排列, 两头留下相等的空白;
- `spread`: 把多余的空白均匀地分布在盒子中间及两侧。

TeXhackers note: 实际这三种方式分别对应 `\leaders`、`\cleaders`、`\xleaders`。

`\atleastfiller`

```
\atleastfiller {<dim expr>}
\atleastfiller [{<filler key-val>}] {<dim expr>}
```

填充的长度至少为 $\langle dim\ expr \rangle$, 太短的将自动断行。

我能吞下玻璃而不伤身体, I can eat glass, it dosen't hurt me.% 例 25
`\atleastfiller[cdotted=1em]{5cm}` 断行。

我能吞下玻璃而不伤身体。`\atleastfiller[cdotted=1em]{5cm}` 不断。

我能吞下玻璃而不伤身体, I can eat glass, it dosen't hurt me.
 断行。

我能吞下玻璃而不伤身体。 不断。

`\breakablefiller`

```
\breakablefiller
\breakablefiller [{<filler key-val>}]
```

可自动断行的 filler。默认使用空格填充。

`\framebox[3cm]{可断}` `\breakablefiller[cdotted=1em]` `\framebox[3cm]{模式}`。例 26

`\framebox[7cm]{可断}` `\breakablefiller[cdotted=1em]` `\framebox[7cm]{模式}`。

可断 模式。

可断

. 模式。

下例展示了制作多行填充的例子。

`\newcommand\filllines[4][]{\% filler key-val, before, lines, after` 例 27
`#2\filler[#1]%`
`\Replicate{#3-1}{\break \rule{0pt}{0.7\baselineskip}\filler[#1]}%`
`#4\par}}`

我能吞下玻璃而不伤身体, I can eat glass, it dosen't hurt me.
`\filllines{\linespread{2}\selectfont}{3}{. \hspace*{1em}}`

我能吞下玻璃而不伤身体, I can eat glass, it dosen't hurt me.
`\filllines[color=red,dotted]{\linespread{2}\selectfont}{3}{. \hspace*{1em}}`

% 整行
`\filllines [raise=-.5ex]{\linespread{2}\selectfont \noindent\strut}{3}{`
`\hspace*{1em}}`
 ↪ 整行。`\hspace*{1em}}`

我能吞下玻璃而不伤身体, I can eat glass, it dosen't hurt me. _____

_____。

我能吞下玻璃而不伤身体, I can eat glass, it dosen't hurt me.

.....

.....。
整行。

2.4.3 多栏文字

CuS_TE_X 中排版多栏文字有两种方式, 本节描述其中一种, 使用 `multicol` 宏包实现。另一种方式见第 5.7 节。

在 `multicol` 中, 可以使用

```
\begin{multicols}{col}
...
\end{multicols}
```

例 28

来排版多栏文字。本模块对其进行了简单的封装, 使得可以通过键值方式来设置相关变量。

关于每个内部变量的详细用法, 可以参考 `multicol` 宏包文档。

```
\startmulticolumns [multicolumns key-val]
<content>
\stopmulticolumns
```

```
\startmulticolumns
\stopmulticolumns
```

将 `<content>` 以多栏排版。

```
columns|cols = {<整数表达式>}
```

初始值: 2

```
multicolumns/columns
multicolumns/cols
```

设置多栏数。也可不必写出键名, 只写数字。

可用的栏数为 1–20, 如果 `multicolumns/adjP25` 为 `true`, 则可用的栏数为 1–10。

```
outer-sep = {<skip expr>}
```

初始值: 12.0pt plus 4.0pt minus 3.0pt

```
multicolumns/outer-sep
```

设置 `\multicolsep`, 即多栏文字与上下文的间距。

```
column-sep|sep = {<length>}
```

初始值: 10pt

```
multicolumns/column-sep
multicolumns/sep
```

设置 `\columnsep`, 即多栏文字两栏的间隙。

```
first-minimal = {<pre length>}
```

初始值: 50pt

```
multicolumns/first-minimal
```

```
last-minimal = {<post length>}
```

初始值: 20pt

```
multicolumns/last-minimal
```

如果多栏开始的那一页不足 `<pre length>`, 则多栏将在新的一页开始。

如果多栏结束的那一页不足 `<post length>`, 则多栏将在新的一页结束。

`first-minimal` 设置 `\premulticols`, `last-minimal` 设置 `\postmulticols`。

```
heading = {<content>}
```

```
multicolumns/heading
```

设置在多栏文字之前的横跨所有栏的文字。可以使用 `\sectionP30` 等。它与其后的多栏文字保持在同一页。

```
rule-width = {<length>}
```

初始值: 0pt

```
multicolumns/rule-width
```

```
rule-color = {<color>}
```

```
multicolumns/rule-color
```

```
rule-color = [color mode] {<color>}
```

设置 `\columnseprule`、`\columnseprulecolor`, 即多栏间竖线的宽度和颜色。

<div> <div>multicolumns/flush</div> <div>multicolumns/aligned</div> <div>multicolumns/ragged</div> <div>multicolumns/not-aligned</div> </div>	<div>flush aligned</div> <div>ragged not-aligned</div> <div>控制多栏文字的尾部是否对齐。分别设使用 \flushcolumns 和 \raggedcolumns。初始为 aligned, 将使各栏头部和尾部的基线尽量对齐。</div>	<div>不可设置值</div> <div>不可设置值</div>
<div> <div>multicolumns/balanced</div> <div>multicolumns/not-balanced</div> </div>	<div>balanced = <true false></div> <div>not-balanced</div> <div>在末页文字的处理上, 有两种方式, 一种为文字尽量往上排, 而将下方留空, 这也是默认的方式; 另一种为文字尽量往左排 (右排), 而将右边 (左边) 留空, 也就是将空白留在末尾的几栏上。前者为 balanced, 后者为 not-balanced。</div>	<div>初始值: true</div> <div>不可设置值</div>
<div> <div>multicolumns/columns*</div> <div>multicolumns/cols*</div> </div>	<div>columns* cols* = <{<栏数>}></div> <div>它在设置栏数的同时还设置 not-balanced。</div> <div>注意 columns 并未决定使用 balanced 还是 not-balanced。</div>	
<div> <div>multicolumns/swap-column</div> <div>multicolumns/enable-swap-column</div> <div>multicolumns/disable-swap-column</div> </div>	<div>swap-column = <true false></div> <div>enable-swap-column</div> <div>disable-swap-column</div> <div>控制是否在使用了 twoside 文档类选项时, 偶数页逆序输出各栏。</div> <div>enable-swap-column 用于启用此功能, disable-swap-column 用于禁用此功能。</div>	<div>初始值: false</div> <div>不可设置值</div> <div>不可设置值</div>
<div> <div>multicolumns/framed</div> <div>multicolumns/framed-options</div> <div>multicolumns/framed-options+</div> </div>	<div>framed = <none fbox ...></div> <div>framed-options = <{<options>}></div> <div>控制多栏文字整体用哪种盒子框住, 多栏文字仍然可以分页。framed-options 为可能的选项。</div> <div>默认仅提供 fbox 这个可选值, 表示用 \fbox 框住。其它库可能提供额外的值。如 tcb 库提供 tcbox 值 (第 5.2.1 小节), box 库提供 lfbox 值 (第 5.7.2 小节)。</div> <div>使用此选项时, 最好把 multicolumns/overflow[ⓘ]_{P25} 设置为 0pt, 否则可能无法正常分页。</div>	

adj = {true false}	初始值: false
adj*	不可设置值
adj-inner = {⟨内侧长度⟩}	
adj-outer = {⟨外侧长度⟩}	
shorten = {⟨长度⟩} {⟨内侧长度⟩, ⟨外侧长度⟩}	
widen = {⟨长度⟩} {⟨内侧长度⟩, ⟨外侧长度⟩}	

multicolumns/adj
multicolumns/adj*
multicolumns/adj-inner
multicolumns/adj-outer
multicolumns/shorten
multicolumns/widen

多栏文字还可以通过调整边距来调整总的文字宽度。adj 用于启用或关闭此功能。adj* 设置 adj=true 及 multicolumns/not-balanced_{P.24}。

adj-inner 的 ⟨内侧长度⟩ 和 adj-outer 的 ⟨外侧长度⟩ 分别调整文字的内侧和外侧边距。正值表示向文字内调整 (总的文字宽度减少), 负值表示向文字外调 (总的文字宽度增加)。它们自动设置 adj 为 true。

shorten 的 ⟨内侧长度⟩ 和 ⟨外侧长度⟩ 表示内侧和外侧缩短指定值, 如果只有一项, 则内侧和外侧均缩短该长度。如果任意一项为空, 则相当于设置为 0pt。它自动设置 adj 为 true。

widen 的 ⟨内侧长度⟩ 和 ⟨外侧长度⟩ 表示内侧和外侧伸长指定值, 如果只有一项, 则内侧和外侧均伸长该长度。如果任意一项为空, 则相当于设置为 0pt。它自动设置 adj 为 true。



wrap-box = {true false}	初始值: false
-------------------------	------------

multicolumns/wrap-box

当增加了多栏文字的宽度时, 如果这些文字还被放在一个盒子中, 则可能会造成 Overfull, 设置本选项为真, 则不会出现 Overfull。

这种情况常见于把这些文字放在浮动体中, 因此, 在浮动体中自动设置该选项为真。

一般情况无需特别设置该选项。

addto-baselineskip = {⟨length⟩}

multicolumns/addto-baselineskip

设置 \multicolbaselineskip。

tolerance = {⟨int expr⟩}	初始值: 9999
pretolerance = {⟨int expr⟩}	

multicolumns/tolerance
multicolumns/pretolerance

设置 \multicoltolerance、\multicolpretolerance。

collectmore = {⟨int expr⟩}

multicolumns/collectmore
multicolumns/minrows
multicolumns/unbalance
multicolumns/column-badness
multicolumns/final-column-badness

设置 collectmore, minrows, unbalance, columnbadness, finalcolumnbadness 计数器。

top-fuzz = {⟨dim expr⟩}	初始值: 0pt
bottom-fuzz = {⟨dim expr⟩}	初始值: 2pt

multicolumns/top-fuzz
multicolumns/bottom-fuzz

设置 \multicolovershoot、\multicolundershoot。

v-fuzz = {⟨length⟩}

multicolumns/v-fuzz
multicolumns/h-fuzz

v-fuzz 设置 top-fuzz 和 bottom-fuzz。h-fuzz 设置 \hfuzz。

overflow = {⟨dim expr⟩}	初始值: 12pt
-------------------------	-----------

multicolumns/overflow

设置 \maxbalancingoverflow。

```
multicolumns/left-to-right
multicolumns/LR
multicolumns/right-to-left
multicolumns/RL
```

```
left-to-right|LR
right-to-left|RL
```

不可设置值
不可设置值

使用 `\LRmulticolcolumns` 或 `\RLmulticolcolumns`。默认为 `left-to-right`。

2.4.4 额外增加文字的宽度

基于多栏文字的功能, 提供了 `extrawidth` 环境, 用以为部分文字增加额外的宽度, 以及 `fullpagewidth` 环境, 用于输出占整个纸张宽的文字。它们都有多栏环境的功能。

```
\startextrawidth
\stopextrawidth
```

```
\startextrawidth {<inner extra width>} {<outer extra width>}
\startextrawidth [<multicolumns key-val>] {<inner extra width>} {<outer extra width>}
<material>
\stopextrawidth
```

`<material>` 的内侧伸长 `<inner extra width>`, 外侧伸长 `<outer extra width>`, 这两个值可以为空, 表示不增加额外的宽度。

```
\startfullpagewidth
\stopfullpagewidth
```

```
\startfullpagewidth
\startfullpagewidth [<multicolumns>] <inner shrink>, <outer shrink>
<material>
\stopfullpagewidth
```

`<material>` 占整个纸张的宽度。内侧留白 `<inner shrink>`, 外侧留白 `<outer shrink>`, 它们都可以为空, 表示不留白。

2.4.5 旋转的盒子

CuS_TE_X 封装了 `rotating` 宏包, 提供了旋转的盒子。

```
\startrotate
\stoprotate
\Rotate
```

```
\startrotate [<rotate key-val>]
<content>
\stoprotate
\Rotate [<rotate key-val>] {<content>}
```

将 `<content>` 旋转显示。

旋转的盒子有两种方式, 一种为保留旋转后的盒子的大小, 另一种设置旋转后的盒子大小为 0。

```
rotate/turn
rotate/nospaceturn
rotate/rotate
rotate/sideways
```

```
turn = {<number>}
rotate|nospaceturn = {<number>}
sideways
```

不可设置值

将盒子旋转 `<number>` 度。一般是逆时针旋转。

`turn` 使用第一种方式, `rotate` 使用第二种方式。`sideways` 相当于 `turn=90`。

`\startrotate ... \stoprotate` 默认使用 `turn`, `\Rotate` 默认使用 `rotate`。

```
rotate/float
rotate/float*
rotate/figure
rotate/figure*
rotate/table
rotate/table*
```

```
float = {<float type>}
float* = {<float type>}
figure
figure*
```

不可设置值

不可设置值

将 `<content>` 看作在浮动环境 `<float type>` 内, 并将其旋转 90 度。旋转后的内容占据一整个页面。

带 `*` 类似于带 `*` 的浮动环境。

也可不写出 `float` 或 `float*` 键名, 直接写 `<float type>` 或 `<float type>*`。

§ 5 背景, bgfg 模块

bgfg 是对 shipout 钩子的简单封装。

关于“钩子”机制, 第 3.1 节对其作了简单的介绍, 更详细的用法请参考: lthooks-doc.pdf。

本手册前几页的水印使用如下代码实现:

```
\background + [./watermark]{%
  \rotatebox{45}{\color{gray!30}\fontsize{100}{0}%
    \sffamily \CusTeX}}
```

例 29

```
% 使用如下代码即可删除此背景
\removebackground[./watermark]
```

```
\foreground    {<content>}
\foreground +  {<content>}
\foreground (<位置>) {<content>}
\foreground [<hook label>] {<content>}
\foreground + (<位置>) [<hook label>] {<content>}
```

```
\foreground
\background
```

将 <content> 放置在前景或背景中。

- <content> 为要放置的内容, 该内容将完整地嵌于页面内;
- <位置> 是 <content> 要放置的位置, 为两个字符, 前一个为水平位置; 后一个为垂直位置。水平位置包括: 左 (l)、右 (r)、内侧 (i)、外侧 (o); 垂直位置包括: 顶部 (t)、底部 (b); 它们的组合也就是 *layout* 的四个角。此外还有一个 cm, 它表示 *layout* 的正中心, 这也是默认值; 还可以置为空, 此时 <content> 中可以使用 \put 命令指定文字的位置, 纸张左上角为原点;
- <hook label> 为 hook 的 label; 此参数与 <位置> 的先后位置可交换;
`\foregroundF27` 默认为 ./fg, `\backgroundF27` 默认为 ./bg;
 关于 hook label 的作用, 请参考第 3.1 节或 lthooks-doc.pdf;
- 默认情况下 <content> 仅添加到当前页, 使用 + 可将 <content> 添加到往后各页。不带 + 的, <hook label> 无效。

本例展示在页面样式的代码中为本页增加背景文字:

```
\background(lt){\textcolor{red}{\LARGE\CusTeX}}
```

例 30

如本页左上角所示。

除了上述两个命令外, 还提供了两个设置背景图片的命令。

```
\foregroundpicture    {<图片>}
\foregroundpicture +  {<图片>}
\foregroundpicture *  {<图片>}
\foregroundpicture + * {<图片>}
\foregroundpicture + * (<位置>) [<hook label>] {<图片>}
\foregroundpicture + * [<图片选项>] (<位置>) [<hook label>] {<图片>}
```

```
\foregroundpicture
\backgroundpicture
```

将 <图片> 添加到前景或背景中。

+, <位置>、<hook label> 的用法如前所述。其中 <位置> 只能是关于位置的值。

不带 * 的图片被伸缩到与 *layout* 同宽高。而带星号的则仅缩放宽度, 保持纵横比例不变。

也可直接使用 `\backgroundP.27` 放置背景图片。

```
\background{ob}{\includegraphics[width=\marginparwidth]{ctanlion.pdf}}
```

例 31

如本页底部图片所示。

```
\removeforeground
\removebackground
```

```
\removeforeground
\removeforeground [{hook label}]
```

移除前景或背景。

§ 6 索引, index 模块

CuTeX 提供了添加多个索引的方法。并且能够自动编译索引文件。

目前暂未提供 `splitidx` 宏包的功能, 也不与其兼容。

应该与 `glossaries` 宏包兼容。

```
\newindextype
\setupindex
```

```
\newindextype [{index keys}] {<index type>}
\setupindex   [{<index type list>}] {<index keys>}
```

`\newindextypeP.28` 创建一个新的索引 `<index type>`。`\setupindexP.28` 配置 `<index type list>`。`<index type>` 可以使用 `\empty` 作为名称, 此时它的名称为空。

```
\makeindex
```

```
\makeindex
\makeindex [{index keys}]
\makeindex [{index keys}] [{index type}]
```

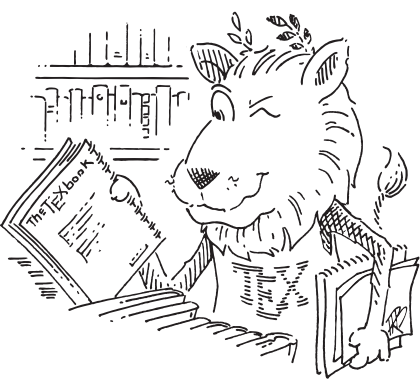
L^AT_EX 原有的接口。默认创建名称为空的索引。

`<index keys>` 不同于 CuTeX 中其它的键值选项, 仅具有类似的接口。

- `filename`: 索引文件名, 一般以 `idx` 结尾, 如果未设置, 则为:
`\jobname<index type>.idx`;
- `output`: 编译后的索引文件, 一般以 `ind` 结尾, 如果未设置, 则将 `filename` 的后缀修改为 `ind` 作为输出文件名;
- `name`: 如果设置, 则应与 `<index type>` 一致;
- `title`: 索引的标题, 如 `\indexname`;
- `program`: 编译索引的可执行程序; 如 `makeindex`、`makeglossaries`;
- `options`: 编译索引时的选项, 索引文件名和输出文件名将自动添加;
- `exec`: 终端中实际编译索引执行的代码, 如果未设置, 则组合 `program` 及 `options`;
- `auto`: 布尔值, 是否自动编译索引文件;
- `multi`: 多栏选项 (`<multicolumns key-val>`);
- `heading*`: 标题命令, 如 `\chapter[numbering=false]`;
- `init`: 索引开头的初始化设置; 索引文件不存在时不会执行;
- `prologue`: 索引开头的文字; 索引文件不存在时不会输出。

```
\newindextype[
  filename=\jobname.docusage.idx,
  output=\jobname.docusage.ind,
```

例 32




```
exec={makeindex -s gind.ist -o \jobname.docusage.ind
↪ \jobname.docusage.idx},
title={代码索引}, heading*={\section},
multi={2, ragged, sep=1em, outer-sep=0pt},
auto=true
]{docusage}
```

```
\setindexinit {<code>}
\setindexinit [<index type>] {<code>}
\setindexprologue {<content>}
\setindexprologue [<index type>] {<content>}
```

```
\setindexinit
\setindexprologue
```

设置索引开头的初始化设置、设置索引开头的文字。只要使用了 `\printindexP29`, 当索引不存在时它们也会执行或输出。

默认设置名称为空的索引。

在初始化代码中还可以重定义索引环境 `theindex`。

```
\index {<index item>}
\index [<index type>] {<index item>}
\printindex
\printindex [<index keys>]
```

```
\index
\printindex
```

`\indexP29` 添加索引项 `<index item>` 到索引 `<index type>` 中, 默认添加到名称为空的索引中; `\printindexP29` 输出索引, 以 `name` 键标识要输出的索引, 否则输出名称为空的索引。`<index keys>` 为前述的键。

§ 7 文档结构, struct 模块

`struct` 模块提供了创建目录和章节标题的方法。参考了 `titlesec`, `titletoc`, `ctex-heading`, `etoc` 等宏包的实现, 并自动阻止加载这些宏包。

章节标题样式的设置与 `ctexheading` 宏包也即 C_TE_X 文档类的接口基本一致, 但扩充了几个选项, 并且可以定义新的标题。

```
\definetitle {<title command>} {<title key-val>}
\definetitle {<title command>} [<title class>] {<title key-val>}
```

```
\definetitle
```

定义新的章节标题命令 `<title command>`, 以 `<title class>` 作为基准类。

标题的使用方式见下方预定义的几个章节标题命令。

标准的 L_AT_EX book 类中, 章节标题可分为三种, 一种是以 `\partP30` 为代表的, C_US_TE_X 将其命令为 `page` 类, 一种是以 `\chapterP30` 为代表的, C_US_TE_X 将其命名为 `top` 类, 另一种则是以 `\sectionP30` 为代表, C_US_TE_X 将其命名为 `normal` 类。¹ 另外还有 `free`, `wrap` 类, 详细用法见第 4.1 节。

T_EXhackers note: 定义章节标题时, 为了使得 `plain` 格式和 `template` 格式的目录正常工作, 需要分别定义 `\l@<title command name>` 和 `templatecbl` 对象类型对应的实例。详见第四章。

本模块预先定义了一些章节命令, 它们与 `ctexbook` 文档类的效果基本一致。

¹实际上, 这些名称基本沿用了 `titlesec` 宏包的名称。

<code>\part</code>	<code>\part {<标题>}</code>
<code>\chapter</code>	<code>\part [⟨list entry⟩] {<标题>}</code>
<code>\section</code>	<code>\part [⟨title key-val⟩] {<标题>}</code>
<code>\subsection</code>	<code>\part [⟨title key-val⟩] [⟨list entry⟩] {<标题>}</code>
<code>\subsubsection</code>	<code>\part - {<标题>}</code>
<code>\paragraph</code>	<code>\part - [⟨list entry⟩] {<标题>}</code>
<code>\subparagraph</code>	<code>\part - [⟨title key-val⟩] {<标题>}</code>
	<code>\part - [⟨title key-val⟩] [⟨list entry⟩] {<标题>}</code>
	<code>\part * {<标题>}</code>
	<code>\part * [⟨title key-val⟩] {<标题>}</code>

与标准的章节标题命令相比, 增加了 `⟨title key-val⟩` 可选项, 用于暂时修改样式。

`⟨标题⟩` 为实际显示的标题, `⟨list entry⟩` 为目录、页眉等内容中的文字, 它在带星号的命令中无效。

带 `-` 的命令不会增加计数器的值, 也不会显示编号, 其它的与不带 `-` 的命令一致。相当于 `mode=nonumber`, 见 [mode](#)_{P.34}。

若要修改它们的样式, 一般仅需使用下述的 `\setuptitle`_{P.30} 修改键值选项, 而无需重新定义它们。

<code>\setuponetitle</code>	<code>\setuponetitle {⟨title level⟩} {⟨key-val⟩}</code>
<code>\setuptitle</code>	<code>\setuponetitleq {⟨title⟩} {⟨key-val⟩}</code>
	<code>\setuptitle {⟨title key-val⟩}</code>
	<code>\setuptitle [⟨title level list⟩] {⟨title key-val⟩}</code>

设置标题的样式。`⟨title level⟩` 为章节标题层级, 如: `chapter`、`section`、`1`。`⟨title⟩` 为章节标题名, 而非标题命令 (`\chapter`、`\section`)。

<code>\setupnexttitle</code>	<code>\setupnexttitle {⟨title keys⟩}</code>
	<code>\setupnexttitle + {⟨title keys⟩}</code>
	<code>\setupnexttitle ? {⟨title keys⟩}</code>
	<code>\setupnexttitle ? + {⟨title keys⟩}</code>

设置标题的选项 `⟨title keys⟩`, 但只作用于接下来的那个标题, 这个标题完成后, 会清空它。

带 `+` 的, 把 `{⟨title keys⟩}` 附加到之前所保存的选项之后。带 `?` 的, 仅会设置已知的选项, 当选项未定义时会忽略它们。当为某些标题定义了额外的键, 但不确定接下来使用的标题是否有这些键时, 它很有用。

以下几节描述了章节标题中可用的键值选项, 它们均可以被设置, 但并不一定在所有的标题类中都有效。这里的 `...` 代表各章节标题命令的名称。

2.7.1 初始化设置

初始化设置选项可以在导言区修改任意次, 但不可在正文中设置。

<code>title/.../number-from</code>	<code>number-from = {⟨计数器⟩}</code>
<code>title/.../number-within</code>	<code>number-within = {⟨计数器⟩}</code>
<code>title/.../number-without</code>	<code>number-without = {⟨计数器⟩}</code>

设置章节命令的计数器。

`number-from` 设置章节命令所使用的计数器, 默认为使用自己的计数器, 该计数器与章节命令同名。

`number-within` 设置章节计数器随该计数器的递增而清零。`number-without` 取消对应的设置。

```
number-format = {\code}
label-format  = {\code}
```

```
title/.../number-format
title/.../label-format
```

设置输出的数字的格式（即 `\the<section>`）或交叉引用的格式。

```
level = {\整数或层级名称}
```

```
title/.../level
```

设置章节标题的层级。将影响是否对标题进行编号。

在正文中展开为此前的标题的名称。例如现在它的值为 “subsection”。

```
\CurrentTitleName *
```

在正文中展开为此前的标题的 *level*。在没有使用标题前，它展开为 -10001。

```
\CurrentTitleLevel *
```

例如，现在它的值为 2。

展开为章节标题的使用次数。

```
\thetitlecount *
```

`\thetitlecountP31` 为到目前为止的使用次数，`\PreviousTitleCountP31` 为上一次运行时章节标题的使用次数或为 0。

```
\PreviousTitleCount *
```

2.7.2 编号

```
\setsecnumdepth {\整数或层级名称}
```

```
\setsecnumdepth
```

设置对章节标题进行编号的层次数。可以是整数或层级名称。

CuS_TE_X 预先定义了一些层级名称，如表 2.2 所示。

层级	-1	0	1	2	3
名称	part	chapter	section	subsection	subsubsection
层级	4	5	4	5	
名称	paragraph	subparagraph	sub3section	sub4section	

表 2.2: 层级名称

```
numbering = {true|false} 初始值: true
```

```
title/.../numbering
```

控制是否对不带星号的命令编号。当此选项设置为 false 时，除了不带编号，其余功能均与正常标题一致。

注意，章节是否编号还受到 secnumdepth 计数器的控制，可以通过上述的 `\setsecnumdepthP31` 命令来设置。例如，对于 `\sectionP30` 而言，其默认的章节层级为 1（对于预定义的几个章节标题，其默认的章节层级与同名层级名称的层级相同，见表 2.2，可通过 `levelP31` 键来修改层级）。因此 `\sectionP30` 会被编号当且仅当 secnumdepth 不小于 1，且其 numbering 键为 true，并且使用不带星号的章节命令。

```
name = {\名字的前半部分},{\名字的后半部分}
name = {\名字的前一部分}
name-pretto = {\pre to}
name-appto = {\app to}
```

```
title/.../name
title/.../name-pretto
title/.../name-appto
```

设置章节的名字。所谓“章节的名字”，可以分为前后两部分，即章节编号前后的词语，两个词之间用一个半角逗号分开；也可以只有一部分，表示只有章节编号之前的名字。

name-pretto 把 `\pre to` 附加到名字之前，name-appto 把 `\app to` 附加到名字之后。

例如，如果想在某一章节前添加 *，只需使用 `name-pretto=*`，或 `name-pretto=\llap{* }`。

```
title/.../number
```

```
number = {⟨数字输出命令⟩}
```

设置章节编号的数字输出格式。⟨数字输出命令⟩通常是对应章节编号计数器的输出命令，如 `\thesection` 或 `\zhnum{chapter}` 之类的。

`number` 选项定义的不会控制对章节计数器的交叉引用。在引用计数器时，记录在 L^AT_EX 辅助文件中的仍然是原来的定义，但可以通过 `label-formatP.31` 修改。

2.7.3 格式

和 C_TE_X 宏集一样，CuS_TE_X 也提供了提供了 `numberformat`, `nameformat`, `titleformat`, `format` 这几个选项用来控制章节标题的格式。它们的作用范围如图 2.4 所示。

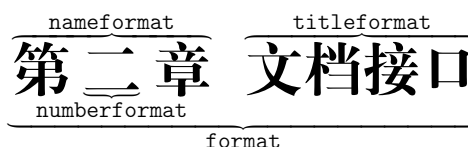


图 2.4: `numberformat`, `nameformat`, `titleformat`, `format` 几个选项的作用范围

```
title/.../format
title/.../format+
title/.../nameformat
title/.../nameformat+
title/.../numberformat
title/.../numberformat+
title/.../titleformat
title/.../titleformat+
```

```
format = {⟨格式代码⟩}
```

```
format+ = {⟨格式代码⟩}
```

设置相应部分的格式。带 + 的用于在原有的格式后增加代码。注意，+ 与选项之间不能留有空白，不能写成 `format += {..}`，以下同。

```
title/.../aftername
title/.../aftername+
title/.../aftertitle
title/.../aftertitle+
```

```
aftername = {⟨code⟩}
```

```
aftername+ = {⟨code⟩}
```

用于将 ⟨code⟩ 插入到相应的部分之后。带 + 的用于在原有的格式后增加代码。

```
title/.../pagestyle
```

```
pagestyle = {⟨pagestyle⟩}
```

`page` (如 `\partP.30`) 和 `top` (如 `\chapterP.30`) 标题类还可以设置该标题所在页的页面样式。在 `normal` 标题类中无效。值为空时，不会修改页面样式。

关于页面样式的相关内容，见第 2.3.7 小节。

2.7.4 间距和缩进

```
title/.../runin
```

```
runin = ⟨true|false⟩
```

初始值: `false`

用于指定是否是标题与其后的文字排在同一行。仅对 `normal` 类有效。

```
title/.../hang
```

```
hang = ⟨true|false⟩
```

初始值: `false`

设置是否对章节标题实施悬挂缩进（缩进的宽度为名字宽度和 `indent` 选项设置的宽度之和）。设置该选项为 `true` 时必须恰当地设置 `aftername` 选项。

若设置了 `runin` 为 `true`，则该选项无意义。

```
title/.../indent
```

```
indent = {⟨缩进间距⟩}
```

设置章节标题本身的首行缩进。如果这缩进值是相对单位，则缩进间距的大小是相对于 `format` 中指定的字号大小。

```
beforeskip = {\skip expr}
```

设置章节标题前后左右的垂直间距。若 `runin` 选项为 `true`, 则设置的是水平间距。

其中, 左右间距只在某些类中有效。

```
title/.../beforeskip
title/.../afterskip
title/.../leftskip
title/.../rightskip
```

```
fixskip = <true|false>
```

初始值: **false**

默认情况下, 章节标题除了由 `beforeskip` 和 `afterskip` 选项设置的垂直间距外, 还会有其它一些多余的间距, `fixskip` 用于指定是否抑制这些多余的间距。

```
title/.../fixskip
```

```
ensureskip = <true|false>
```

初始值: **false**

使用某些标题类时, 标题如果出现在新的一页, `beforeskip` 可能并不一定准确, 可以使用此选项以确保 `beforeskip` 有准确的值。

```
title/.../ensureskip
```

```
break = {\code}
```

```
break+ = {\code}
```

`break` 选项用于控制章节标题与之前正文的分隔关系。一般用于设置是否在标题之前分页或者设置行间罚点。

例如, 若当前页剩余高度小于正文高度的一半时, 则另起一页输出 `\section`^{P30} 标题:

```
\usepackage{needspace}
\setuptitle [section] { break = \Needspace{0.5\textheight} }
```

例 33

```
title/.../break
title/.../break+
```

```
afterindent = <true|false>
```

`afterindent` 选项用于设置章节标题后首段的缩进。

```
title/.../afterindent
```

2.7.5 浮动体

CuTeX 提供了可以控制本章节内的浮动体位置的接口。

```
float-barrier = <true|false>
```

初始值: **false**

控制是否本章节所属的浮动体可以位于其它章节内, 为 `true` 时, 浮动体不能放在其它章节内。默认情况下浮动体可以放置在其它章节内。

除了使用以上这个选项, 还可以设置浮动体的“边界”。

阻止 `\FloatBarrier`^{P33} 前的浮动体被放置在这个命令的后边。

```
title/.../float-barrier
```

```
\FloatBarrier
```

2.7.6 杂项

```
beforerecord = {\code}
```

在写入辅助文件之前执行一些必要的设置。例如, 如果需要保存 `xpos` 和 `ypos`, 必须要先执行 `\pdfsavepos`, 则可以使用该键。

```
title/.../beforerecord
title/.../beforerecord>
title/.../beforerecord<
```

```
tocline = {\格式定义}
```

定义章节标题在目录文件中的格式, `<格式定义>` 有两个参数: 参数 #1 是 `part`, `chapter` 等名字, 参数 #2 是标题内容。

初始值是: `\titlenumberline{#1}#2`。其含义为, 若标题没有名字, 则不输出编号。

```
title/.../tocline
```

`title/.../mark`

`mark = {\mark code}`

写入标记文本。`\mark code` 其后跟一个参数, 在 `\chapterP.30` 中, 为 `\chaptermark`, 在 `\sectionP.30` 中, 为 `\sectionmark`。初始情况下, 不写入标记。

`title/.../properties`
`title/.../properties+`

`properties = {\properties list}`

使用 `\RecordProperties` 向 `.aux` 文件写入辅助信息。

写入的 *label* 名为 `cus.title.\thetitlecountP.31`。

可以使用 `\RefProperty{cus.title.\thetitlecountP.31}{\property}` 获得在当前标题下写入的 `\property` 的值。

`title/.../bookmark`
`title/.../bookmark*`
`title/.../bookmark-extra`

`bookmark = {\text}`
`bookmark* = {\text}`
`bookmark-extra = {\text}`
`bookmark-extra = [\bookmark options] {\text}`

设置此条标题在书签中的文字。

`bookmark` 会在 `bookmark` 宏包设置了 `numbered` 选项后, 把数字也加上。`bookmark*` 则直接设置书签为 `\text`。

`bookmark-extra` 额外添加一个书签。用在带星号的命令中可以为增加书签。

`\titleaddinfo`
`\titleremoveinfo`
`\titleremoveallinfo`

`\titleaddinfo {\property} {\value}`
`\titleaddinfo * {\property} {\value}`
`\titleremoveinfo {\property}`
`\titleremoveinfo * {\property}`
`\titleremoveallinfo`
`\titleremoveallinfo *`

为下一个 (或多个) 标题添加 (或移除) 额外的信息, 以 `\property` 标识。

当 `\titleaddinfo` 和 `\titleaddinfo*` 有同一个 `\property` 时, 不带星号的具有更高优先级。

`title/.../style`

`style = {\style}`

设置已有的样式。

`title/.../mode`

`mode = <normal|nonumber|starred|...>`

设置标题的模式。具体效果取决于标题类的定义。

带星号的标题相当于 `mode=starred`, 但带星号的命令优先级更高, 设置 `mode` 不能取消带星号的效果。

`numbering=true` 相当于 `mode=normal, numbering=false` 相当于 `mode=nonumber`。

`\titleifnamed *`

`\titleifnamed {\有名字时的内容} {\无名字时的内容}`

根据当前章节有无名字展开得到不同内容 (通常是格式命令)。

`\ifintitle *`

`\ifintitle {\在标题中执行} {\不在标题中执行}`

根据是否在标题中执行相应代码。从正要设置键值之前到恢复键值原来的定义之间这段代码都算作在标题中。另见 `\ifincblentryP.34`。

`\ifincblentry *`

`\ifincblentry {\写入目录条目时} {\非写入目录条目时}`

判断是否正在写入目录条目。

每一个标题都有一个对应的 `\titlethe{title}` 命令, 表示当前实际输出的章节标题的名字。如现在 `\titlethechapter` 为“第二章”。

2.7.7 目录

CusTeX 重新实现了目录的制作方式, 将每个目录项看成是一个个数据, 不同于标准的目录制作方式, 因此可能存在与其它宏包不兼容的情况。

在 CusTeX 中, 目录被称为“combined list”, 这是 ConTeXt 中的称呼。

CusTeX 的 `struct` 一共定义了三种格式的目录: `plain`、`template`、`specified`。这三种目录格式可定制程度依次增加, 使用方式也依次更加复杂。

`\enablecombinedlist`^{P35} 启用目录。可用于导言区或文档最开头。如果未使用此命令则其它目录命令不可用。

`\enablecombinedlist`

`from = {\file}` 初始值: `\jobname`

`cbl-setup/from`

设置目录的来源。默认为 `\jobname`, 即来自本文件。不包含文件后缀。

`write = {true|false}` 初始值: `true`
`to = {\file}`

`cbl-setup/write`
`cbl-setup/to`

控制当前主文件是否写入目录, 及写入到哪个文件。如果设置了 `to`, 则写入到 `\file.toc`, 否则写入 `from` 键指定的那个文件。

`cus` 提供了与 `book` 类相似的目录格式, 这种目录称为“plain”格式的目录。

当 `\contentsname`、`\listfigurename`、`\listtablename` 为空时, 对应的 `plain` 格式目录不会附带章节标题, 否则将自动添加一个正常的带序号的章标题。它们可以通过 `\settocdepth`^{P60} 控制输出的层级。

`\tableofcontents` [`\multicolumns keyval`]
`\localtableofcontents` [`\multicolumns keyval`] (`\level`), (`\index`)

`\tableofcontents`
`\listoffigures`
`\listoftables`
`\localtableofcontents`

输出 `plain` 格式的标题、图片和表格目录。

`\localtableofcontents`^{P35} 输出局部的标题目录, 层级为 `\level`, 位置为 `\index`。

`\level` 用于设置局部目录的层级, 例如为 `chapter` 时, 输出本章目录。默认为最近使用的那个标题的层级。

如下代码输出本小节的目录:

<code>\renewcommand\contentsname{}</code> % 移除自动添加的标题	例 34
<code>\localtableofcontents(section)</code> % 输出本节目录, 本 <code>\section</code> 也会输出	
<hr/>	
§ 7 文档结构, struct 模块	29
2.7.1 初始化设置	30
2.7.2 编号	31
2.7.3 格式	32
2.7.4 间距和缩进	32
2.7.5 浮动体	33
2.7.6 杂项	33
2.7.7 目录	35

`\multicolplaincombinedlist`

`\multicolplaincombinedlist` [*<multicolumns key-val>*] {<title>} {<cbl type>}

`\multicolplaincombinedlist`^{P₃₆} 输出 plain 格式的多栏目录, 该目录的类型是 <cbl type>, 并以 <title> 作为标题。如果 <title> 为空, 则不输出标题。<multicolumns key-val> 设置多栏选项, 如果栏数为 1, 则相当于默认的单栏目录。

`\multicollocalplaincombinedlist`

`\multicollocalplaincombinedlist` [*<multicolumns keyval>*]
{<title>} {<cbl type>} {<min>} {<max>}

输出多栏局部目录。局部目录从 <min> 开始, 到 <max> 终止。如果 <title> 为空, 则不输出标题。

实际上, `\localtableofcontents`^{P₃₅} 根据 <level> 和 <index> 隐式计算了 <min> 和 <max>。

可以使用 `\getcbllevelrange`^{P₃₈} 获得对应位置小节目录的范围 <min>、<max>。

cus 提供了设置用键值对目录进行简易调整的目录格式, 称为“template”格式的目录。

当 `\contentsname`、`\listfigurename`、`\listtablename` 为空时, 对应的 template 格式目录不会附带章节标题, 否则将自动添加一个正常的带序号的章标题。它们可以通过 `\settocdepth`^{P₆₀} 控制输出的层级。

`\templatetoc`
`\templatelof`
`\templatelot`

`\templatetoc`
`\templatetoc` [*<templatecbl keys>*] [*<multicolumns key>*] (<level>,<index>)

输出“template”格式的目录。<templatecbl keys> 用于局部修改目录的输出。

`\templatecombinedlist`

`\templatecombinedlist` {<type>} {<title>}
`\templatecombinedlist` [*<templatecbl keys>*] [*<multicolumns key>*] {<cbl type>}
(<level>,<index>) {<title>}

template 格式目录的完整形式。

下面列出 <templatecbl keys> 可用的键值选项, 它们只能在 `\templatetoc`^{P₃₆} 等命令内部使用, 不能在外部的 `\cussetup`^{P₂} 中设置。

`templatecbl/hide`
`templatecbl/show`

`hide =` {<cbl level list>}

隐藏或显示 <cbl level list> 指定的目录层级。

`templatecbl/*`

`*` = {<template keys>}

把 <template keys> 应用到当前目录类型的所有目录层级。<template keys> 是下文将要说明的模板键。

`templatecbl/no-parent`

`no-parent`

不可设置值

移除局部目录的第一项。

对于一个局部目录, 如输出本章目录 (chapter 层级), 不仅会输出本章的小节写入的目录条目, 还会输出章 (即 `\chapter`^{P₃₀}) 写入的目录内容, 使用此选项将不会输出章的条目。

对于没有定义的 *templatecbl key*, 首先会判断它是不是以 * 开头, 若是, 则用该键的值修改这个 * 后面的大括号中的那些目录实例。否则, 尝试将它解释为一个 *templatecbl* 目录对象的一个实例名, 然后使用该键给出的值修改此实例。如果也不是一个实例名, 则会报错。关于目录对象和目录实例, 见下文。

<pre> \renewcommand\contentsname{} % 移除自动添加的标题 \templatetoc[no-parent,*{subsection}={ignore=true}, section={ignore=false}](chapter) % 输出本章目录, 但隐藏 \chapter 和 ↳ \subsection </pre>		例 35
§ 1	ltx 模块	2
§ 2	util 模块	7
§ 3	页面布局, layout 模块	9
§ 4	盒子和填充, box 模块	17
§ 5	背景, bgfg 模块	27
§ 6	索引, index 模块	28
§ 7	文档结构, struct 模块	29
§ 8	buffer 模块	40

template 格式的目录, 使用 `xtemplate` 宏包处理键值。一个模板的对象类型用于规定的参数的数量, 一个对象类型可以定义多个模板, 每个模板有自己的处理代码和可用的键, 在模板的基础上为一些键设置固定的值就构成了对象类型的实例。对象类型的实例总是唯一的, 而不论它是设置了哪个模板的键。

template 格式的目录, 其对象类型为 `templatecbl`, 可使用 7 个参数, 为目录条目所保存的值。默认的模板为 `by_name`, 实例为目录条目的层级, 如 `part`、`chapter` 等等。用户可以自己定义新的模板, 详见第 4.3 节。关于如何修改定义模板、修改实例等见第 3.2 节或 `xtemplate.pdf`。

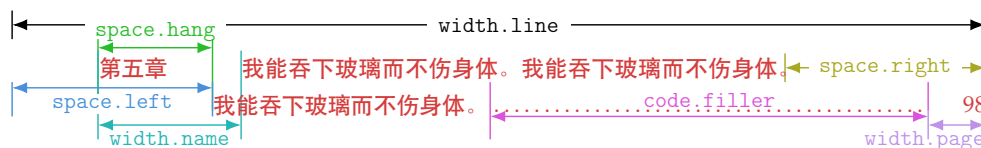


图 2.5: template 格式目录的变量

`by name` 模板可用的键如下 (注意它们不能用 `\cussetup` 设置, 且每个键都必须给出值, 不能省略, 包括布尔值的 `true` 也不能省, 这是 `xtemplate` 所限):

space.before 如果不是为 `0pt`, 且在垂直模式, 则在该目录项之前添加指定的间距; 结果保存于 `\tmcb1@beforeskip` 寄存器之中;

space.left 左侧间距; 结果保存于 `\tmcb1@leftskip` 寄存器之中;

space.right 右侧间距, 默认情况下, 页码会嵌入到右侧空白之中; 结果保存于 `\tmcb1@right` 寄存器之中;

space.hang 第一行额外的缩进, 正值向左侧缩进, 负值向右侧缩进; 结果保存于 `\tmcb1@hang dim` 寄存器之中; 默认为 `width.name` 的值;

space.indent 除第一段外, 每段首行额外的缩进; 结果保存于 `\tmcb1@indent dim` 寄存器之中; 默认为 `space.hang` 的值;

penalty.before 增加 `penalty`; 结果保存于 `\tmcb1@beforepenalty int` 寄存器之中;

format 设置标题前的数字 (即 `name`)、标题文字 (即 `title`)、引导线 (即 `leader`) 和页码 (即 `page`) 的格式; 结果保存于 `\tmcb1@format` 宏之中;

format.name 设置 `name` 的格式, 若样式没有覆盖, `format` 所设仍会生效; 结果保存于 `\tmcb1@nameformat` 宏之中;

format.title 设置 title 的格式, 若样式没有覆盖, format 所设仍会生效; 结果保存于 \tmcb1@titleformat 宏之中;

format.page 设置 page 的格式, 若样式没有覆盖, format 所设仍会生效; 结果保存于 \tmcb1@pageformat 宏之中;

width.name 设置 name 的宽度, 具体效果如何取决于 code.name; 结果保存于 \tmcb1@namewidth dim 寄存器之中;

width.title 设置 title 的宽度, 具体效果如何取决于 code.title; 结果保存于 \tmcb1@titlewidth dim 寄存器之中;

width.page 设置 page 的宽度, 具体效果如何取决于 code.page; 结果保存于 \tmcb1@pagewidth dim 寄存器之中; 默认为 \@pnumwidth;

width.line 设置 \tmcb1@linewidth dim 寄存器;

ignore 是否忽略该层级的目录项; 结果保存于 \tmcb1@ignore, 可使用 \ifodd 或 \bool_if:NTF 等判断;

name.width 同 width.name;

name.format 同 format.name;

name.code 同 code.name;

title.width 同 width.title;

title.format 同 format.title;

title.code 同 code.title;

page.width 同 width.page;

page.format 同 format.page;

page.code 同 code.page;

leader.code 同 code.leader;

leader.sep 设置引导线的内容的间距; 最终结果保存于 \tmcb1@leadersep 宏之中, 默认为 4.5; 当大于 5000 时, 不使用引导线, 而使用 \filler₁₉;

leader.options 引导线使用 \filler₁₉ 时的选项; 结果保存于 \tmcb1@leaderoptions 宏之中; 默认为 space;

leader.content 设置引导线的内容; 结果保存于 \tmcb1@leadercontent 宏之中; 默认为西文句点 .;

hyper.range 设置超链接要包含的内容, 可选值为 name、title、page、none 和 all, 可多选; 最终结果保存于 \tmcb1@hyperrange int 寄存器之中, 其中 name 将其加 1, title 将其加 2, page 将其加 4; 默认为 page;

hyper.code 同 code.hyper;

code 将该目录层级要执行的代码替换为它, 可使用 7 个参数, 即目录条目保存的值; 将结果保存在 \tmcb1@code 命令之中;

code.name 将 name 执行的代码替换为它, 结果保存在 \tmcb1@name 命令之中;

code.title 将 title 执行的代码替换为它, 结果保存在 \tmcb1@title 命令之中;

code.leader 将 leader 执行的代码替换为它, 结果保存在 \tmcb1@leader 命令之中;

code.page 将 page 执行的代码替换为它, 结果保存在 \tmcb1@page 命令之中;

code.hyper 将添加超链接要执行的代码替换为它, 可使用 4 个参数, 分别为 name、title、leader 和 page 要执行的代码, 结果保存在 \tmcb1@hyper 命令之中;

中;

code.before 在目录条目之前执行代码;

code.after 在目录条目之后执行代码, 默认为 `\par`; 注意每个目录项不会自动另起一行, 依靠它来另起新行。

在 `code.name`、`code.title` 等代码中可以使用上述保存的寄存器 (或宏), 若在其它地方使用它们则不保证其有正确结果。此外, 在这些代码中还可以使用下述宏 / 命令:

`\tmcblthetype` 当前使用的目录类型;

`\tmcbltheclass` 当前目录层级名;

`\tmcblthelevel` 当前目录层级对应的整数;

`\tmcbltheinfo` 此目录条目保存的额外信息, 为一个 L^AT_EX3 `prop` 变量, 即 `\titleaddinfo`^{P.34} 添加的值;

`\tmcblthename` `name` 的文字, 可能为空;

`\tmcblthetitle` `title` 的文字;

`\tmcblthepage` `page` 的文字, 即 `\thepage` 的值;

`\tmcbltheanchor` 超链接的锚点, 可能为空; 可直接使用 `\tmcbl@hyperlink` 或 `\tmcbl@hyperlinkbox` 添加超链接;

`\tmcbl@code@` `{\type}{\count}{\infos}{\level int}{\list entry}{\thepage}{\anchor}` 默认的 `code` 实现;

`\tmcbl@name@` 默认的 `code.name` 实现;

`\tmcbl@title@` 默认的 `code.title` 实现;

`\tmcbl@leader@` 默认的 `code.leader` 实现;

`\tmcbl@page@` 默认的 `code.page` 实现;

`\tmcbl@hyper@` `{\name}{\title}{\leader}{\page}` 默认的 `code.hyper` 实现;

`\tmcbl@hyperlink` `{\text}` 为 `text` 添加链接到正文标题处的超链接;

`\tmcbl@hyperlinkbox` `{\text}` 为 `text` 添加链接到正文标题处的超链接;

仿照 `etoc` 宏包, 提供了类似于 `\etocsetstyle` 的命令, 这种目录称为 “specified” 目录。

```
\tocsetstyle {\level list} {\list start} {\block start} {\block item}
{\block finish} {\list finish}
\lotsetstyle {\list start} {\block start} {\block item} {\block finish} {\list finish}
\lofsetstyle {\list start} {\block start} {\block item} {\block finish} {\list finish}
\specifiedtoc
\localspecifiedtoc
\localspecifiedtoc (<level>,<index>)
```

```
\tocsetstyle
\lotsetstyle
\lofsetstyle
\specifiedtoc
\specifiedlot
\specifiedlof
\localspecifiedtoc
```

设置章节目录、表格目录、图片目录的样式。`\specified...` 则用于输出指定的目录。`\localspecifiedtoc`^{P.39} 用于输出局部章节的目录。

详细用法见第四章。

`\SaveSpecifiedCombinedListStyle`

`\SaveSpecifiedCombinedListStyle {⟨cmd⟩} {⟨style code⟩}`

把 $\langle style code \rangle$ 中设置的样式保存在 $\langle cmd \rangle$ 中。这些样式是由 `\tocsetstyleP39`、`\SetSpecifiedCombinedListStyleP40` 等命令设置的。当要保存其它内容到 $\langle cmd \rangle$ 中时, 可以使用 `\+` 命令。

相比把 `\tocsetstyleP39` 等直接保存到 $\langle cmd \rangle$ 中, 在实际使用 $\langle cmd \rangle$ 时, 还会有其它处理, 而使用 `\SaveSpecifiedCombinedListStyleP40` 保存则不会有多余的处理。

类似 `\keys_precompile:nnN` 之于 `\keys_set:nn`。

如下代码将一段样式设置保存在 `\mysavedtocstyle` 中, 还保存了 `\newcommand{...}`。

```
\SaveSpecifiedCombinedListStyle \mysavedtocstyle {
  \tocsetstyle
    {chapter,section,subsection}
    {\begin{description}}
    {}
    {\item[\mytoclabel]\tocthetitle\quad\toclink{\tocthepage}\par}
    {}
    {\end{description}}
  \+{\newcommand{\mytoclabel}
    {\tocifnamed{\tocthename}{\rule{1ex}{1ex}}}}
}
```

例 36

`\SetSpecifiedCombinedListStyle`
`\SpecifiedCombinedList`
`\LocalSpecifiedCombinedList`

```
\SetSpecifiedCombinedListStyle [⟨type list⟩] {⟨level list⟩}
  {⟨list start⟩} {⟨block start⟩} {⟨block item⟩} {⟨block finish⟩} {⟨list finish⟩}
\SpecifiedCombinedList [⟨cbl type⟩]
\LocalSpecifiedCombinedList [⟨cbl type⟩]
\LocalSpecifiedCombinedList [⟨cbl type⟩] (⟨level⟩,⟨index⟩)
```

完整形式。详细用法见第四章。

关于目录的内部数据结构, 见第 3.6 节和第四章。

关于章节标题和目录的详细用法和样例, 见第四章。

§ 8 buffer 模块

未完成。[TODO]

第三章 编程接口

本章描述 CuSTeX 提供的编程接口。

`\CUSProvideLibrary`
`\CUSProvideExplLibrary`

```
\CUSProvideLibrary    {⟨库名⟩} [⟨描述⟩]
\CUSProvideExplLibrary {⟨库名⟩} {⟨日期⟩} {⟨版本⟩} {⟨描述⟩}
```

提供库文件。库文件名必须是: “cus.library.⟨库名⟩.tex”。

```
\CUSLibraryDelayedUntil <{package}>
\CUSLibraryDelayedUntil * <{}>
\CUSLibraryDelayedUntil * <{package}>
```

```
\CUSLibraryDelayedUntil
```

用于库文件的第一行，将本库标记为需要延迟加载。

标记本库延迟至加载了 $\langle package \rangle$ 之后再加载，带星号的命令将库延迟至加载了文档类之后再加载。若有其它库依赖使用了本命令的库，则它们都至少会延迟至 $\langle package \rangle$ 之后再加载。

```
\CUSDependency <{key-val}>
\CUSDependency * <{key-val}>
```

```
\CUSDependency
```

处理库依赖。

若用在主文件中或使用带星号的版本，所需的依赖会立刻被加载。用在库文件中，依赖可能会延迟加载。

```
package = <{comma list}>
module   = <{comma list}>
library  = <{comma list}>
disable  = <{comma list}>
```

```
dependency/package
dependency/module
dependency/library
dependency/disable
```

\backslash CUSDependency_{P.41} 可用的键值选项。

T_EXhackers note: 只能以如下顺序执行上述三个命令： \backslash CUSLibraryDelayedUntil_{P.41}、 \backslash CUSDependency_{P.41}、 \backslash CUSProvideExplLibrary_{P.40}（或 \backslash CUSProvideLibrary_{P.40}）。在每个库文件中，它们最多只能使用一次，且必须用在库文件开头。若要在库文件中间使用它们，则应该将之后的内容拆分至新的库文件中。

一个库文件可能会被使用多次，但在 \backslash CUSProvideExplLibrary_{P.40}（或 \backslash CUSProvideLibrary_{P.40}）之后的内容只会执行一次。

```
\CUSLoadLibrary [<选项>] <{库名}> [<日期>]
\CUSPassOptionsToLibrary <{选项}> <{库名}>
```

```
\CUSLoadLibrary
\CUSPassOptionsToLibrary
```

尽量不要使用 \backslash CUSLoadLibrary_{P.41}，它可能不会正确处理库依赖。

```
\text_md5five_hash:n <{text}>
```

```
\text_md5five_hash:n *
```

先使用 \backslash text_expand:n 展开 $\langle text \rangle$ ，然后计算它的 MD5 值。

§ 1 L^AT_EX 2_ε的钩子机制

本节简述 L^AT_EX 2_ε的钩子机制。更详细的说明见 lthooks-doc.pdf。

“钩子”（hook）是在命令或环境的定义中可以添加其它代码的位置。

```
\UseHook <{hook}>
\UseHookWithArguments <{hook}> <{number}> <{arg1>} ... <{argn>}
```

```
\UseHook
\UseHookWithArguments
```

在命令或环境中执行 $\langle hook \rangle$ 。

```
\AddToHook <{hook}> [<label>] <{code}>
\AddToHookWithArguments <{hook}> [<label>] <{code}>
```

```
\AddToHook
\AddToHookWithArguments
```

将 $\langle code \rangle$ 添加到 $\langle hook \rangle$ 中，标记为 $\langle label \rangle$ 。 $\langle hook \rangle$ 可以未被定义。

如果未指定 $\langle label \rangle$ ，则使用默认的 label。如果 \backslash AddToHook_{P.41} 用在宏包或文档类中，它就是宏包或文档类名，否则，它是 top-level。

`\RemoveFromHook`

`\RemoveFromHook {<hook>} [<label>]`

移除标记了 $\langle label \rangle$ 的 $\langle hook \rangle$ 。若 $\langle label \rangle$ 未指定, 则使用默认的 `label`。

`\AddToHookNext`

`\AddToHookNextWithArguments``\ClearHookNext`

`\AddToHookNext {<hook>} {<code>}``\AddToHookNextWithArguments {<hook>} {<code>}``\ClearHookNext {<hook>}`

将 $\langle code \rangle$ 添加到 $\langle hook \rangle$ 的下一调用中。在正常的 $\langle hook \rangle$ 代码执行完毕后再执行 $\langle code \rangle$ 。

§ 2 L^AT_EX 2_ε 的模板机制

内容、排布内容的方式以及具体的配置项构成了一个“文档”。其中内容为用户输入, 排布内容的方式一般是一个环境或命令, 具体的可配置项就是命令的某些参数或者可供修改的变量。其中用户输入是未知的, 而排布内容的方式和修改配置的方式则是固定的, 且有多种实现方式, 模板机制就是其中一种。

本节内容简要介绍 L^AT_EX 2_ε 的模板机制, 更详细的用法请参考 `xtemplate.pdf`。目前暂以单独的宏包提供此功能, L^AT_EX 2_ε Released 2024-06-01 版本将把此机制合并入内核。

对象类型 (object type) 是对用户输入的抽象描述, 它限定了用户输入参数的数目, 一般也限定了顺序。例如, 对于一个章节标题来说, 用户输入参数可能是: 短标题、完整标题。

`\DeclareObjbectType`

`\DeclareObjectType {<object type>} {<no. of args>}`

定义一个对象类型 $\langle object type \rangle$ 。

模板规定了如何处理可配置的项和用户输入的参数, 亦即排布内容的方式。

`\DeclareTemplateInterface`

`\DeclareTemplateInterface {<object type>} {<template>} {<no. of args>} {<key list>}`

L^AT_EX 2_ε 模板机制以键值对的方式处理可配置的项, 本命令用于限定用户可配置哪些项, 以及用何种方式保存这些项, 以及这些项的默认值。

`\DeclareTemplateCode`

`\DeclareTemplateCode {<object type>} {<template>} {<no. of args>}`
`{<key bindings>} {<code>}`

定义可配置项与哪一命令绑定起来, 以及排布内容的方式。

`\UseTemplate`

`\UseInstance {<object type>} {<template>} {<settings>} <arguments>`

把配置项设置为某些固定的值, 并离开使用它。

`\SetTemplateKeys`

`\SetTemplateKeys {<object type>} {<template>} {<keyvalues>}`

用于在模板的代码中修改实例的值。

`\EditTemplateDefaults`

`\EditTemplateDefaults {<object type>} {<template>} {<new defaults>}`

修改模板的默认值。只会应用到新创建的实例。

用某一模板把可配置项设置为某些特定的值就成为了实例。


```
\DeclareInstance {<object type>} {<instance>} {<template>} {<parameters>}
\DeclareInstanceCopy {<object type>} {<instance2>} {<instance1>}
```

```
\DeclareInstance
\DeclareInstanceCopy
```

使用 $\langle parameters \rangle$ 创建 $\langle template \rangle$ 的实例。或创建 $\langle instance_1 \rangle$ 的复本 $\langle instance_2 \rangle$ 。

```
\UseInstance {<object type>} {<instance>} <arguments>
```

```
\UseInstance
```

使用实例。

```
\EditInstance {<object type>} {<instance>} {<new values>}
```

```
\EditInstance
```

修改已有的实例。

```
\IfInstanceExistTF {<object type>} {<instance>} {<true code>} {<false code>}
```

```
\IfInstanceExistTF
\IfInstanceExistT
\IfInstanceExistF
```

判断实例是否存在。

§ 3 ltx 模块

```
\cus@filename {<filename>}
```

```
\cus@filename *
```

自动处理 $\langle filename \rangle$ 中的特殊符号（活动字符和双引号等），展开为一个文件名。

目录（文件夹）的分隔符必须为 $/$ ，即是在 Windows 系统上。

```
\cus@getgraphicsname <cmd> {<filename>}
```

```
\cus@getgraphicsname
```

将 $\langle cmd \rangle$ 设置为 $\langle filename \rangle$ 对应的图片文件名，若图片不存在则为 $\backslash relax$ 。

它会自动查找 $\backslash setgraphicspath$ ₄ 设置的路径，且可以自动补全文件扩展名。

需要用户自行加载 **graphicx** 宏包。

```
\cus_get_graphics_full_name:nN {<filename>} <tl>
\cus_get_graphics_full_name:nNTF {<filename>} <tl> {<true code>} {<false code>}
```

```
\cus_get_graphics_full_name:nN
\cus_get_graphics_full_name:nNTF
```

将 $\langle tl \rangle$ 设置为 $\langle filename \rangle$ 对应的图片文件名，若图片不存在则为 $\backslash q_no_value$ 。

它会自动查找 $\backslash setgraphicspath$ ₄ 设置的路径，且可以自动补全文件扩展名。

需要用户自行加载 **graphicx** 宏包。

§ 4 util 模块

3.4.1 交叉引用、超链接和书签

L^AT_EX 的 $\backslash label$ 可以标记位置用于交叉引用，**hyperref** 宏包还提供了超链接的功能，**bookmark** 宏包则提供了书签功能。本模块封装了其中的某些命令，但不会自动加载这些宏包（如果没有加载所需的宏包，这些命令不会报错，只是被忽略掉了），需要用户自行加载或使用 CuSTeX 提供的宏包加载机制来加载。

```
\labelinfo          *
\cus_label_info:nn *
```

```
\labelinfo {<info>} {<label>}
```

获取 $\langle label \rangle$ 的相关信息。 $\langle label \rangle$ 可以为空，代表最近写入的那个 label。

可获得的信息 $\langle info \rangle$ 为：

- name, $\langle label \rangle$ 的值，若 $\langle label \rangle$ 不存在则为 $\backslash c_novalue_tl$ 的值；
- page, 获得 $\langle label \rangle$ 所在页的 $\backslash thepage$ 值，若 $\langle label \rangle$ 不存在则为 0；
- ref, 获得 $\langle label \rangle$ 中的第一个数据项，也就是 $\backslash ref\{\langle label \rangle\}$ 显示的内容，若 $\langle label \rangle$ 不存在则为 $\backslash c_novalue_tl$ 的值，可使用 $\backslash IfNoValueTF$ 或 $\backslash tl_if_novalue:nTF$ 判断；
- anchor, 获得链接到 $\langle label \rangle$ 所在位置的锚点，若 $\langle label \rangle$ 不存在或未加载 **hyperref** 则为 Doc-Start；
- pageanchor, 获得链接到 $\langle label \rangle$ 所在页的锚点，若 $\langle label \rangle$ 不存在或未加载 **hyperref** 则为 Doc-Start。

注意：anchor 和 pageanchor 不会将 **hyperref** 的 $\backslash HyperDestNameFilter$ 命令考虑在内，如果需要，可以使用 **hyperref** 的 $\backslash hyperget\{anchor\}\{\langle label \rangle\}$ 和 $\backslash hyperget\{pageanchor\}\{\langle label \rangle\}$ 。

```
\cus_newlabel_now:nnnnnn
\cus_newlabel_now:xxxxxx
\cus_newlabel_shipout:nnnnnn
\cus_newlabel_shipout:xxxxxx
\cus_newlabel_shipout_x:nnnnnn
\cus_newlabel_shipout_x:xxxxxx
```

```
\cus_new_label_now:nnnnnn {<label>} {<ref data>} {<thepage>}
{<current label name>} {<anchor>} {<extra>}
```

写入一个 $\langle label \rangle$ 。

$\langle label \rangle$ 、 $\langle thepage \rangle$ 、 $\langle current label name \rangle$ 、 $\langle anchor \rangle$ 总是立即展开。

它们不会执行 label 钩子。

作为一个例子， $\backslash label\{\#\}$ 类似于 $\backslash cus_newlabel_shipout:nnnnnn\{\#\}\{\backslash @currentlabel\}\{\backslash thepage\}\{\backslash @currentlabelname\}\{\backslash @currentHref\}\{\backslash @kernel@reserved@label@data\}$

```
\cus_make_target:n
\cus_make_unique_target:n
```

```
\cus_make_target:n {<target>}
\cus_make_unique_target:n {<target>}
```

$\backslash cus_make_target:n$ ^{P.44} 以 $\langle target \rangle$ 为名，创建一个锚点。 $\langle target \rangle$ 必须唯一。锚点位置自动升高 $\backslash normalbaselineskip$ 。

$\backslash cus_make_unique_target:n$ ^{P.44} 创建一个锚点，其名以 $\langle target \rangle$ 为前缀，由一个共享的计数器保证这个锚点名唯一，每调用一次，这个计数器都会自增。

加载了 **hyperref** 宏包才有效。

```
\g_cus_anchor_tl
\cus_gset_next_anchor_name:n
\cus_gset_next_anchor_raise:n
```

```
\cus_gset_next_anchor_name:n {<name>}
\cus_gset_next_anchor_raise:n {<dim>}
```

$\backslash g_cus_anchor_tl$ ^{P.44} 保存了最近一个锚点的名称，它是只读的。相当于 $\backslash @currentHref$ 。

$\backslash cus_gset_next_anchor_name:n$ ^{P.44} 设置下一个锚点的名称。 $\langle name \rangle$ 被立刻展开。

$\backslash cus_gset_next_anchor_raise:n$ ^{P.44} 设置下一个锚点要升高的高度。 $\langle dim \rangle$ 立即被计算。直接使用的 $\backslash Hy@raisedlink$ 不会受影响。

它们也会影响 **hyperref** 宏包中其它创建锚点的命令。


```
\cus_ref_label:nn {<label>} {<text>}
\cus_ref_target:nn {<target>} {<text>}
\cus_ref_label_box:nn {<label>} <material>
\cus_ref_target_box:nn {<target>} <material>
```

```
\cus_ref_label:nn
\cus_ref_target:nn
\cus_ref_label_box:nn
\cus_ref_target_box:nn
```

将 <text> 或 <material> 链接到 <label> 或 <target>。

<text> 可以断行，但不能包含特殊文本。<material> 中可以包含特殊文本，如 verbatim，仅能在特殊的位置断行。另见 \cus_ref_label_shbox:nn^{P.55}、\cus_ref_target_shbox:nn^{P.55}。

<material> 可以是 {<horizontal mode material>}，正如 \hbox {<horizontal mode material>} 那样，也可以有 <box specification>。左右括号可以是隐式的。

加载了 hyperref 宏包才有效。

```
\ExplSyntaxOn
\cus_ref_label_box:nn { sec:module-util } \bgroup\verb|本节开始|\egroup 或
\cus_ref_label:nn { sec:module-util } {
  ↳ 链接到本节开始，但是是很长很长很长很长很长很长的可以断行的文字。 }
\ExplSyntaxOff
```

例 37

本节开始或链接到本节开始，但是是很长很长很长很长很长很长的可以断行的文字。

把文字的颜色改为对应的颜色。

注意：使用 \hypersetup{linkcolor=red} 等修改的颜色仅在 PDF 阅读器中显示，在打印时不会显示，而使用上述 6 个命令以及 \color、\color_select:n 等修改的颜色会在打印时显示。

```
\cus@colorfile
\cus@colorlink
\cus@colorcite
\cus@colorurl
\cus@colorrun
\cus@colormenu
```

```
\cus_bookmark:nn {<options>} {<bookmark>}
\cus_gset_next_bookmark_text:n {<bookmark>}
```

```
\cus_bookmark:nn
\cus_gset_next_bookmark_text:n
```

设置书签。或设置下一个书签的文字。

\cus_bookmark:nn^{P.45} 是对 \bookmark 的封装。加载了 bookmark 宏包才有效。

```
\cus_if_pdfstring:TF {<true>} {<false>}
```

```
\cus_if_pdfstring_p: *
\cus_if_pdfstring:TF *
```

判断是否在书签等 PDF 字段内。

```
\cus_hyper_anchor:n {<destination name>}
\cus_hyper_anchor_start:n {<destination name>} <content>
\cus_hyper_anchor_stop:
\cus_hyper_link:nnn {<context>} {<destination name>} {<link text>}
\cus_hyper_link_start:nn {<context>} {<destination name>} <content>
\cus_hyper_link_stop:
\cus_hyper_link_file:nnn {<link text>} {<filename>} {<destname>}
\cus_hyper_link_url:nn {<link text>} {<url>}
\cus_hyper_link_launch:nnn {<filename>} {<link text>} {<Parameters>}
\cus_hyper_link_name:nn {<action>} {<link text>}
```

```
\cus_hyper_anchor:n
\cus_hyper_anchor_start:n
\cus_hyper_anchor_stop:
\cus_hyper_link:nnn
\cus_hyper_link_start:nn
\cus_hyper_link_stop:
\cus_hyper_link_file:nnn
\cus_hyper_link_url:nn
\cus_hyper_link_launch:nnn
\cus_hyper_link_name:nn
```

对驱动文件提供的基础命令的封装，必须加载 hyperref 宏包才有效。其中最后两个仅在使用了通用驱动（generic driver）才有效（即使用了 \DocumentMetadata 的特性）。

它们仅创建锚点（或创建超链接），不会设置任何格式。

```
\cus_gset_next_page_label:n
\cus_gset_page_label:n
\cus_gset_page_label_code:n
\cus_reset_page_label_code:
```

```
\cus_gset_next_page_label:n {<page label>}
\cus_gset_page_label:n {<page label>}
\cus_gset_page_label_code:n {<code>}
\cus_reset_page_label_code:
```

这些命令用于设置在阅读器中显示的页码。 $\langle page label \rangle$ 一般包含 `\thepage` 或 `\arabic{page}` 等内容。`\cus_gset_next_page_label:n`^{例 38} 相当于 `hyperref` 宏包的 `\thispdfpagelabel` 命令，用于设置本页的 page label。

$\langle code \rangle$ 带有一个参数，使用 `\renewcommand`、`\def`、`\tl_set:Nn` 等命令设置这个参数方可改变 page label。注意， $\langle code \rangle$ 执行于 `shipout/before` 钩子中，此时 page 计数器已经递增，但 `\g_shipout_readonly_int` (`\ReadonlyShipoutCounter`)、`\g_shipout_totalpages_int` 还未改变。

加载了 `hyperref` 宏包且 `pdfpagelabels` 为真才有效。

例如，下例为阅读器中显示的本页页码加上 SP. 前缀。

```
\ExplSyntaxOn
\cus_gset_next_page_label:n { SP.\thepage }
% 相当于下面这段代码
% \cus_gset_page_label_code:n
% {
%   \tl_set:Nn #1 { SP.\thepage }
%   \cus_reset_page_label_code:
% }
\ExplSyntaxOff
```

例 38

3.4.2 向前查找和收集内容

L^AT_EX3 的以 `peek` 为模块名的命令可以用于向前查找、检测和分析记号，`collectbox` 宏包提供了向前收集内容存进盒子的功能。本模块也实现了类似的命令。

```
\cus_peek_verbatim:nw
```

```
\cus_peek_verbatim:nw {<tokens1>} {<balanced tokens>}
\cus_peek_verbatim:nw {<tokens1>} <token> <tokens> <token>
```

以纯文字的形式向后收集一段代码，然后把它们放在一个组中，然后把 $\langle tokens \rangle$ 放到它前面。 $\langle balanced tokens \rangle$ 或两个相同 $\langle token \rangle$ 之间的 $\langle tokens \rangle$ 不能作为命令的参数。

```
\cus_peek_act:nnnnn
```

```
\cus_peek_act:nnnnn
{<normal>} {<space>} {<group begin>} {<group end>} {<else>}
```

类似于 `\peek_N_type:TF`，但对后面的记号执行对应的分支。这个记号仍然保留下来。

$\langle else \rangle$ 的情况一般是后面的记号为 `\outer` 宏。

3.4.3 分析记号

$\langle token list \rangle$ 可能含有特定的模式，本模块提供了分析某些特定模式的命令。

```
\cus_if_head_int:p:n *
\cus_if_head_int:nTF *
```

```
\cus_if_head_int:nTF {<tl>} {<true code>} {<false code>}
```

测试 $\langle tl \rangle$ 是否以数字起始。

```
\ExplSyntaxOn
1. \cus_if_head_int:nTF { 2022 } { T } { F }
```

例 39

```
\ 2. \cus_if_head_int:nTF { +2022 } { T } { F } % 正数
\ 3. \cus_if_head_int:nTF { -2022 } { T } { F } % 负数
\ 4. \cus_if_head_int:nTF { '3746 } { T } { F } % 8进制数
\ 5. \cus_if_head_int:nTF { "7E6 } { T } { F } % 16进制数
\ 6. \cus_if_head_int:nTF { Notnu } { T } { F }
\ 7. \cus_if_head_int:nTF { \par } { T } { F }
\ 8. \cus_if_head_int:nTF { \c_true_bool } { T } { F } % \char
\ 9. \cus_if_head_int:nTF { \c_one_int } { T } { F } % int (count)
\ 10. \cus_if_head_int:nTF { \tracingmacros } { T } { F }
\ 11. \cus_if_head_int:nTF { \the\tracingmacros } { T } { F } % 展开为整数
\ 12. \cus_if_head_int:nTF { \baselineskip } { T } { F }
\ 13. \cus_if_head_int:nTF { \number\baselineskip } { T } { F } % 展开为整数
\ExplSyntaxOff
```

1.T 2.T 3.T 4.T 5.T 6.F 7.F 8.T 9.T 10.F 11.T 12.F 13.T

```
\cus_arg_to_keyval_apply:nnN {<key name>} {<arg>} {<function>}
\cus_arg_to_keyval_apply:nnn {<key name>} {<arg>} {<tokens>}
```

```
\cus_arg_to_keyval_apply:nnN
\cus_arg_to_keyval_apply:nnn
```

判断 $\langle arg \rangle$ 是否为键值对；如果不是则把 $\langle key name \rangle$ 作为键名， $\langle arg \rangle$ 作为值，形成一个键值对。再把这键值放在 $\langle function \rangle$ （或 $\langle tokens \rangle$ ）的后面。

```
\cus_if_keyval_apply:nNN {<arg>} {<true function>} {<false function>}
\cus_if_keyval_apply:nnn {<arg>} {<true tokens>} {<false tokens>}
```

```
\cus_if_keyval_apply:nNN
\cus_if_keyval_apply:nnn
```

首先检查 $\langle arg \rangle$ 是否为键值对，如果是，则使用 $\langle true \rangle$ 分支，否则使用 $\langle false \rangle$ 分支。再把 $\langle arg \rangle$ 处理后的结果放在 $\langle function \rangle$ （或 $\langle tokens \rangle$ ）的后面。

```
\cus_if_keyval_variable:nNnn {<arg>} {<variable>} {<true code>} {<false code>}
```

```
\cus_if_keyval_variable:nNnn
```

把处理后的 $\langle arg \rangle$ 保存到 $\langle variable \rangle$ 中， $\langle code \rangle$ 可以使用这个 $\langle variable \rangle$ 。

判断是否为键值对的方式和 `ltxcmd` 的 `= spec` 一样，详见 `usrguide.pdf`。

```
\ExplSyntaxOn
\cs_new_protected:Npn \fiicmd #1
{
  \cus_if_keyval_variable:nNnn {#1} \l_tmpa_tl
  { Y [ \tl_to_str:N \l_tmpa_tl ] }
  { N ( \tl_to_str:N \l_tmpa_tl ) }
}
\ExplSyntaxOff
\ttfamily
\fiicmd{,d=1}
\fiicmd{=,d=1}
\fiicmd{=1}
\fiicmd{=$=1$}
\fiicmd{{=,d=1}}
\fiicmd{}
\fiicmd{ }
\fiicmd{=,}
\fiicmd{a{=},}
```

例 40

Y[,d=1] Y[d=1] Y[=1] N(\$=1\$) N({=,d=1}) N() N() Y[] N(a{=},)

```
\cus_split_bracket_head_default:nn *
```

```
\cus_split_bracket_head:n *
```

```
\cus_split_bracket_head_default:nn {⟨default⟩} {⟨tl⟩}
\cus_split_bracket_head:n {⟨tl⟩}
```

解析 $\langle tl \rangle$ 。判断其是否以一对方括号 ([]) 起始, 若是则将方括号后的内容放入一个集合中 ({ })。方括号不可直接嵌套, 需放入组中。

```
\ttfamily \ExplSyntaxOn
\exp_args:Ne \tl_to_str:n
{
  1. \cus_split_bracket_head:n { }
  \space 2. \cus_split_bracket_head:n { tail }
  \space 3. \cus_split_bracket_head:n { [bracket]tail }
  \space 4. \cus_split_bracket_head:n { [bracket] }
}
\ExplSyntaxOff
```

```
1. [{}]{ } 2. [{}]{tail} 3. [{bracket}]{tail} 4. [{bracket}]{ }
```

例 41

```
\cus_split_bracket_tail_default:nn *
```

```
\cus_split_bracket_tail:n *
```

```
\cus_split_bracket_tail_default:nn {⟨tl⟩} {⟨default⟩}
\cus_split_bracket_tail:n {⟨tl⟩}
```

解析 $\langle tl \rangle$ 。判断其是否以一对方括号 ([]) 结尾, 若是则将方括号前的内容放入一个集合中 ({ })。方括号不可直接嵌套, 需放入组中。

```
\ttfamily \ExplSyntaxOn
\exp_args:Ne \tl_to_str:n
{
  1. \cus_split_bracket_tail:n { }
  \space 2. \cus_split_bracket_tail:n { head }
  \space 3. \cus_split_bracket_tail:n { head[bracket] }
  \space 4. \cus_split_bracket_tail:n { [bracket] }
}
\ExplSyntaxOff
```

```
1. {}[{} ] 2. {head}[{} ] 3. {head}[{}bracket] 4. [{}bracket][{} ]
```

例 42

```
\parserange:nnnN
```

```
\parserange:(nnvN|nneN)
```

```
\parserange:nnN
```

```
\parserange:(nvN|neN)
```

```
\parserange:nnnN {⟨最小值⟩} {⟨最大值⟩} {⟨range list⟩} {⟨sequence⟩}
\parserange:nnN {⟨最大值⟩} {⟨range list⟩} {⟨sequence⟩}
```

解析一个整数列表, 将其保存至 $\langle sequence \rangle$ 中。可使用 \rightarrow 标记连续的范围。若范围的开始为空, 则设它为 $\langle \text{最小值} \rangle$, 若终止为空, 则设它为 $\langle \text{最大值} \rangle$ 。

逆序的范围无效。

如果 $\langle \text{最小值} \rangle$ 被省略, 则设它为 1。

```
\parserange_check:
```

```
\parserange_nocheck:
```

是否检查越界值。

当设置了检查越界值时, 结果的项被限制在 $\langle \text{最小值} \rangle$ 和 $\langle \text{最大值} \rangle$ 之中 (含边界)。

否则, 忽略这一限制, 但逆序的范围仍然无效。

```
\ExplSyntaxOn
\parserange:nnN { 10 } { ->2, 4->7, 8-> } \l_tmpa_seq
\seq_use:Nn \l_tmpa_seq { ,~ } \par

\parserange:nnN { 10 } { -1->2, 4->7, 9->12, 20, 30->32 } \l_tmpa_seq
```

例 43

```
\seq_use:Nn \l_tmpa_seq { ,~ } \par

\parserange_nocheck:
% 不检查越界
\parserange:nnN { 10 } { -1->2, 9->12, 20, 30->32 } \l_tmpa_seq
\seq_use:Nn \l_tmpa_seq { ,~ } \par

\parserange:nnN { 10 } { -1->2, 9->12, 20, 32->30 } \l_tmpa_seq
\seq_use:Nn \l_tmpa_seq { ,~ } \par % 逆序, 无效
\ExplSyntaxOff
```

```
1, 2, 4, 5, 6, 7, 8, 9, 10
1, 2, 4, 5, 6, 7, 9, 10
-1, 0, 1, 2, 9, 10, 11, 12, 20, 30, 31, 32
-1, 0, 1, 2, 9, 10, 11, 12, 20
```

```
\parserange_set_to_int:n {<code>}
```

```
\parserange_set_to_int:n
```

有时，给出的值并非为整数，但可以通过某种方式转为整数。此时可以使用此函数设置转化为整数的方法。<code> 可以使用一个参数，为原始的值。<code> 必须为 f-expandable。

```
\parserange_use_delimiter:n {<delimiter>}
\parserange_use_default_delimiter:
```

```
\parserange_use_delimiter:n
\parserange_use_default_delimiter:
```

设置范围的分隔符为 <delimiter>，默认为 ->。对分隔符的修改应该是局部的。

```
\ExplSyntaxOn
\parserange_use_delimiter:n { - }
\parserange:nnN { 10 } { 0-2, 7-12, 20, 32-30 } \l_tmpa_seq
\seq_use:Nn \l_tmpa_seq { ,~ }
\ExplSyntaxOff
```

例 44

```
1, 2, 7, 8, 9, 10
```

```
\cus_tl_split_braced:nn {<tl_1>} {<tl_2>}
```

```
\cus_tl_split_braced:nn *
```

`\cus_tl_split_braced:nn`⁴⁹ 提取 <tl₁> 和 <tl₂> 的值，将它们分别拆成两个部分，这拆分的四个部分中，第 1、3 部分的长度为 <tl₁> 和 <tl₂> 中长度的较小值。第 2、4 个部分为剩余的值，当中至少有一个为空。这四个部分的每一项都由括号括起来。即，若设 <tl₁> 的长度小于 <tl₂>，则

$$\begin{aligned}
 tl_1 &= a_1, a_2, \dots, a_m \\
 tl_2 &= b_1, b_2, \dots, b_m, b_{m+1}, \dots, b_n \\
 res_1 &= tl_1 = a_1, a_2, \dots, a_m \\
 res_2 &= \text{empty} \\
 res_3 &= b_1, b_2, \dots, b_m \\
 res_4 &= b_{m+1}, \dots, b_n
 \end{aligned}$$

最终结果使用 \unexpanded (\exp_not:n) 包裹起来。

`\cus_tl_split_braced:NNNN`

`\cus_tl_split_braced:NNNN <tl var1> <tl var2> <tl var3> <tl var4>`

`\cus_tl_split_braced:NNNN`_{p50} 提取 `<tl var1>` 和 `<tl var2>` 的值, 将它们分别拆成两个部分, 将这些值保存到 4 个 `<tl var>` 中。其中 `<tl var1>` 和 `<tl var2>` 为前一部分, `<tl var3>` 和 `<tl var4>` 为后一部分, 当中至少有一个为空。

```
\ttfamily \ExplSyntaxOn
\tl_to_str:e { \cus_tl_split_braced:nn {12345} {ABCDEFG} } \par
\tl_set:Nn \l_tmpa_tl { 12345 }
\tl_set:Nn \l_tmpb_tl { ABCDEFG }
\cus_tl_split_braced:NNNN \l_tmpa_tl \l_tmpb_tl \l_tmpc_tl \l_tmpd_tl
[ \tl_to_str:N \l_tmpa_tl ] \par
[ \tl_to_str:N \l_tmpb_tl ] \par
[ \tl_to_str:N \l_tmpc_tl ] \par
[ \tl_to_str:N \l_tmpd_tl ] \par
\ExplSyntaxOff
```

例 45

```
{\1}{2}{3}{4}{5}}{\{A}{B}{C}{D}{E}}{\{F}{G}}
[{\1}{2}{3}{4}{5}]
[{\A}{B}{C}{D}{E}]
[]
[{\F}{G}]
```

3.4.4 杂项

`\cus_if_preamble_p:` *
`\cus_if_preamble:TF` *
`\cus_if_document_p:` *
`\cus_if_document:TF` *
`\cus_if_class_loaded_p:` *
`\cus_if_class_loaded:TF` *
`\cus_if_after_documentclass_p:` *
`\cus_if_after_documentclass:TF` *

`\cus_if_preamble:TF <{true code}> <{false code}>`

测试是否在导言区、正文, 或是否已经加载了文档类, 或是否在 `\documentclass` 后。

`\cus_exp_args:Nd` *
`\cus_exp_args:NNd` *
`\cus_exp_args:Nnd` *
`\cus_exp_last_unbraced:Nd` *
`\cus_exp_last_unbraced:NNd` *
`\cus_exp_last_unbraced:Nnd` *

`\cus_exp_args:Nd <function> <{tokens}>`
`\cus_exp_args:NNd <function> <function> <{tokens}>`
`\cus_exp_args:Nnd <function> <{tokens1>}<{tokens2>}`

展开 `<tokens>` 或 `<tokens2>` 两次。

```
\ExplSyntaxOn
\cus_exp_args:Nd \tl_to_str:n
{ \prg_replicate:nn { 5 } { \CusTeX } }
\ExplSyntaxOff

\CusTeX \CusTeX \CusTeX \CusTeX \CusTeX
```

例 46

```
\ExplSyntaxOn
\cus_exp_last_unbraced:Nd \tl_to_str:n
{ \char_generate:nn { ` \ } { 1 } } \token_to_str:N }
\ExplSyntaxOff

\token_to_str:N
```

例 47

`\cus_exp_args:nw {<spec>} <tl>`

按 $\langle spec \rangle$ 将 $\langle tl \rangle$ 依次展开。 $\langle spec \rangle$ 的长度不限。为 `\exp_args:..` 的增强版，但若 `\exp_args:<spec>` 存在，则不宜使用 `\cus_exp_args:nw`_{P51}。

$\langle spec \rangle$ 为下列之组合：N、c、p、n、n_un、o、o_un、d、d_un、f、f_un、V、V_un、v、v_un、e、e_un、_un、_un 以及由括号括起来的空或空格。忽略未被保护的空格。

空的作用是把该项的内容清除，空格的作用是添加一个空格。带有 `_un` 后缀的，使用时需有括号括起来，其结果不自动添加 `{}`。

两次展开即可得到结果。

```
\ExplSyntaxOn
\tl_set:Nn \l_tmpa_tl { \relax }
\cus_exp_args:Nnd \tl_set:Nn \l_tmpb_tl
{
  \cus_exp_args:nw { c v {~} o {o_un} f { } {e_un} }
  { \l_tmpa_tl } { \l_tmpa_tl } { \l_tmpa_tl } { \l_tmpa_tl }
  { ~ 01 } {ig} { \prg_replicate:nn { 5 } { 0 } }
}
\ttfamily \tl_to_str:N \l_tmpb_tl
\ExplSyntaxOff
```

例 48

```
\l_tmpa_tl {\relax }{ }{\relax }\relax {01}{ }00000
```

注意，`o_un`、`V_un` 和 `v_un` 的行为与 `\exp_last_unbraced:N..[o|V|v]` 的行为并不一致，假设有 `\def\foo#1{#1}`，`\exp_last_unbraced:No \~ \foo {a}` 可以得到 \tilde{a} ，`\cus_exp_args:nw {N{o_un}} \~ \foo {a}` 则会出错。

`\cus_serial_exp_arg:nnw {<spec1m>} {<spec2>} <tl1m> {<tl>}`

依次执行 `\cus_exp_args:nw`_{P51} $\{ \langle spec_{1m} \rangle \langle spec_{2i} \rangle \} \langle tl_i \rangle$ ($1 \leq i \leq \langle spec_2 \rangle$ 的长度)，其中 $\langle spec_{2i} \rangle$ 为 $\langle spec_2 \rangle$ 的第 i 项， $\langle tl_i \rangle$ ($i \geq 2$) 为前一次的展开结果， $\langle tl_1 \rangle$ 为 $\langle tl_{1m} \rangle \{ \langle tl \rangle \}$ 。

```
\ExplSyntaxOn
\cs_set_protected:Npn \foo #1 { \int_eval:n { 1+2+#1 } }
\tl_set:Nn \l_tmpa_tl { \l_tmpb_tl }
\tl_set:Nn \l_tmpb_tl { \foo }
\cus_serial_exp_arg:nnw { N } { o o f } \tl_to_str:n { \l_tmpa_tl { 3 } }
\ExplSyntaxOff
```

例 49

6

`\ExpandArgs {<spec>} <token> <tl>`

按 $\langle spec \rangle$ 将 $\langle tl \rangle$ 依次展开。这是由 L^AT_EX_{2_ε} 提供的命令。本模块进一步增强了它。当 `\exp_args:N<spec>` 存在时，使用它，否则使用 `\cus_exp_args:nw`_{P51} $\{ N \langle spec \rangle \}$ 。

`\cus_use_none_num:nw {<num>} <tl>`

移除 $\langle tl \rangle$ 的前 $\langle num \rangle$ 项。 $\langle tl \rangle$ 必须至少有 $\langle num \rangle$ 项。忽略未被保护的空格。如果 $\langle num \rangle$ 小于等于 0，则什么也不做。

两次展开即可得到结果。

`\cus_exp_args:nw *`
`\cus_exp_args:ew *`

`\cus_serial_exp_args:nnw *`
`\cus_serial_exp_args:eew *`

`\ExpandArgs *`

`\cus_use_none_num:nw *`

例 50

```
\ExplSyntaxOn
\cus_exp_args:NNd \tl_set:Nn \l_tmpa_tl
{ \cus_use_none_num:nw { 5 } 1234567890 }
\tl_to_str:N \l_tmpa_tl
\ExplSyntaxOff
```

67890

```
\cus_exp_num:nN *
\cus_exp_after_num:nwN *
```

```
\cus_exp_num:nN {<num>} <token1>
\cus_exp_after_num:nwN {<num>} <token0> <token1>
```

展开 $\langle token_1 \rangle$ $\langle num \rangle$ 次。如果 $\langle num \rangle$ 小于等于 0，则什么也不做。
两次展开即可得到结果。

```
\cus_tl_use:Nnnn *
\cus_tl_use:Nn *
```

```
\cus_tl_use:Nnnn <tl var> {<separator between two>}
{<separator between more than two>} {<separator between final two>}
\cus_tl_use:Nn <tl var> {<separator>}
```

把 $\langle tl \text{ var} \rangle$ 放在输出流中，每项之间加上正确的 $\langle separator \rangle$ ，类似于 `\clist_use:Nnnn` 和 `\clist_use:Nn`。

忽略未使用 $\{ \}$ 保护的空格。

T_EXhackers note: 最终结果使用 `\unexpanded` (`\exp_not:n`) 包裹起来。

```
\cus_tl_use:nnnn *
\cus_tl_use:nn *
```

```
\cus_tl_use:nnnn {<tl>} {<separator between two>}
{<separator between more than two>} {<separator between final two>}
\cus_tl_use:nn {<tl>} {<separator>}
```

把 $\langle tl \rangle$ 放在输出流中，每项之间加上正确的 $\langle separator \rangle$ ，类似于 `\clist_use:nnnn` 和 `\clist_use:nn`。

忽略未使用 $\{ \}$ 保护的空格。

T_EXhackers note: 最终结果使用 `\unexpanded` (`\exp_not:n`) 包裹起来。

例 51

```
\ExplSyntaxOn
\tl_set:Nn \l_tmpa_tl { ~{xparse}{command}~{is}{~}{not}~{expandable}! }
\tl_set:Nx \l_tmpb_tl { \cus_tl_use:Nnnn \l_tmpa_tl { - } { - } { -and- } }
\ttfamily \tl_to_str:N \l_tmpb_tl
\ExplSyntaxOff
```

xparse-command-is- -not-expandable-and-!

```
\cus_act_case:nnnn *
\cus_act_case_true:nnnn *
\cus_act_case_false:nnnn *
```

```
\cus_act_case_true:nnnn {<tl>} {<processor list>} {<fallback tokens>} {<act args>}
```

$\langle processor \text{ list} \rangle$ 中奇数项为判断函数，偶数项为对应的处理代码，如果判断为真（或为假），则使用对应的处理代码，如果 $\langle processor \text{ list} \rangle$ 中的判断函数都判断为假（或都为真），则使用 $\langle fallback \text{ tokens} \rangle$ 。偶数项的处理代码可以使用 $\langle act \text{ args} \rangle$ 作为参数。

`\cus_act_case:nnnnP.52` 是 `\cus_act_case_true:nnnnP.52` 的另一个名字。

例 52

```
\ExplSyntaxOn
% 判断：(catcode=12) 和 (catcode=11) 在哪种情况下相等
\exp_args:No \cus_act_case_true:nnnn
{ \token_to_str:N : } % \catcode\:=12
{
  { \tl_if_eq:nnTF { : } { } { \tl ~ \use:n } % \catcode\:=11 不相等
    { \token_if_eq_catcode:NNTF : } { cat ~ \use:n } % \catcode\:=11 不相等
    { \str_if_eq:nnTF { : } { } { str ~ \use:n } % string 相等
  }
  { not ~ equal \use_none:n }
  { {eq} }
}
\ExplSyntaxOff
```

str eq

```
\cus_if_lazy_all:nnTF {<tl>} {<test list>} {<true>} {<false>}
```

```
\cus_if_lazy_all:nnTF *
\cus_if_lazy_any:nnTF *
```

判断 $\langle tl \rangle$ 是否满足 $\langle test list \rangle$ 中的所有（或任一）判断，如果是则使用 $\langle true \rangle$ ，否则使用 $\langle false \rangle$ 。

例 53

```
\ExplSyntaxOn
\exp_args:No \cus_if_lazy_any:nnTF { \token_to_str:N : }
{
  { \tl_if_eq:nnTF { : } { } } % 不满足
  { \str_if_eq:nnTF { : } { } } % 满足
}
{ true }
{ false }
\
\exp_args:No \cus_if_lazy_all:nnTF { \token_to_str:N : }
{
  { \tl_if_single_token:nnTF : } % 满足
  { \token_if_eq_charcode:NNTF : } % 满足
  { \token_if_eq_catcode:NNTF : } % 不满足
}
{ true }
{ false }
\ExplSyntaxOff
```

true false

这两个命令主要用于那些不可展开的测试，如 $\backslash\mathrm{tl_if_eq:nnTF}$ 、 $\backslash\mathrm{regex_match:nnTF}$ ，或者测试方式多样，如既要使用 $\backslash\mathrm{token_if_eq_meaning:NNTF}$ 又要使用 $\backslash\mathrm{token_if_eq_charcode:NNTF}$ ，因此不能直接使用 $\backslash\mathrm{token_case_meaning:NnTF}$ 和 $\backslash\mathrm{token_case_charcode:NnTF}$ 。

```
\cus_map_nest_code:Nnnn <map tokens function> {<arg>} {<nest>} {<code>}
\cus_map_nest_variable:NnnNn <map tokens function> {<arg>} {<nest>} <variable>
{<code>}
```

```
\cus_map_nest_code:Nnnn
\cus_map_nest_variable:NnnNn
```

使用 $\langle map tokens function \rangle$ 迭代 $\langle arg \rangle$ ，嵌套 $\langle nest \rangle$ 次。 $\langle map tokens function \rangle$ 为以 $_map_tokens:..$ 结尾的宏，如 $\backslash\mathrm{tl_map_tokens:nn}$ 、 $\backslash\mathrm{seq_map_tokens:Nn}$ 。根据第一个参数的不同， $\langle arg \rangle$ 也要随之改变。

$\langle code \rangle$ 可以使用嵌套的结果，嵌套结果的长度为 $\langle nest \rangle$ ， $\langle code \rangle$ 的执行次数为第一层 $\langle code \rangle$ 执行次数的 $\langle nest \rangle$ 次幂。

例 54

```

\ExplSyntaxOn
\cs_set:Npn \__this_box:n #1
{
  \hbox_set:Nn \l_tmpa_box { \scriptsize #1 }
  \hbox_to_wd:nn { \box_wd:N \l_tmpa_box }
  { % 上面显示十进制数字，下面显示二进制数字
    \oalign {
      \hfil \int_from_bin:n {#1} \hfil \cr
      \box_use_drop:N \l_tmpa_box
    }
  }
}
\cus_map_nest_code:Nnnn \tl_map_tokens:nn { 01 } { 4 }
{ [ \__this_box:n {#1} ] } % 共执行 (len("01"))^4=16 次
\ExplSyntaxOff

```

```

[ 0 ][ 1 ][ 2 ][ 3 ][ 4 ][ 5 ][ 6 ][ 7 ][ 8 ][ 9 ][10][11][12][13][14][15]
0000 0001 0010 0011 0100 0101 0110 0111 1000 1001 1010 1011 1100 1101 1110 1111

```

§ 5 box 模块

box 模块封装了一些环境或命令，用于在编程时使用。另见第 6.1 节。

3.5.1 为宽度固定和宽度可变的内容创建超链接

util 模块提供了创建超链接的命令。本模块则定义了可以为宽度固定和宽度可变的内容创建超链接的命令。

```

\cus_ref_label_width:nnnn
\cus_ref_label_varwidth:nnnn
\cus_ref_target_width:nnnn
\cus_ref_target_varwidth:nnnn

```

```
\cus_ref_label_width:nnnn {<label>} {<vpos>} {<width>} <material>
```

将 <material> 链接到 <label> 或 <target> 的位置。其宽度（或最大宽度）为 <width>，垂直位置为 <vpos>。

另见 `\cus_ref_label_box:nn`_{P45}、`\cus_ref_target_box:nn`_{P45}。

例 55

```

\ExplSyntaxOn
\cs_set:Npn \myparfbox
{
  \collectn_minipage:Nnnnw \l_tmpa_box
  { \fbox { \box_use_drop:N \l_tmpa_box } }
}
\cus_ref_label_width:nnnn { sec:module-box-prog } {b} {3cm}
{ 链接到\par 本节开始 } |
\cus_ref_label_varwidth:nnnn { sec:module-box-prog } {t} {3cm}
{ 链接到\par 本节开始 } |
\cus_ref_label_box:nn { sec:module-box-prog }
{ \myparfbox {b} {3cm} { 链接到\par 本节开始 } }
\ExplSyntaxOff

```

链接到		链接到
本节开始	链接到	本节开始
	本节开始	

另见 `\HyperRef`_{P80}、`\HyperLink`_{P80} 及例 70。

3.5.2 特殊的“水平”盒子

T_EX 的 \hbox 所创建的盒子是受限水平模式 (restricted horizontal mode) 下的盒子。这盒子中的 discretionary、penalty 等项被移除了，当使用 \unhbox 时，在

某些位置无法断行。名为 *Line Breaking in \unhboxed Text* 的 TUGBoat 文章介绍了一种方法，可以解决上述问题。

本模块使用此种想法定义了几个命令。

```
\cus_set_shbox:Nn <box> {<content>}
\cus_set_shbox:Nw <box> <content> \cus_set_shbox_end:
```

```
\cus_set_shbox:Nn
\cus_set_shbox:Nw
\cus_set_shbox_end:
```

设置可正常断行的盒子。注意不允许在 $\langle content \rangle$ 中手动断行。

在 $\text{X}\text{T}\text{E}\text{X}$ 中，断行位置与 $\backslash\text{hbox}$ 无区别，即不会增加额外的断行位置。以下均同。

```
\cus_peek_shbox:Nnw <box> {<code>} <material>
```

```
\cus_peek_shbox:Nnw
```

向后收集可正常断行的盒子。注意不允许在 $\langle content \rangle$ 中手动断行。

```
\cus_use_shbox:N <box>
```

```
\cus_use_shbox:N
```

类似于 $\backslash\text{box_use:N}$ ，使用上述两个命令设置的盒子必须使用这个命令来使用盒子。

注意在使用了 $\backslash\text{cus_set_shbox:Nw}_{\text{P}_{35}}$ 后如果没有使用 $\backslash\text{cus_use_shbox:N}_{\text{P}_{35}}$ ，则不能再使用 $\backslash\text{cus_set_shbox:Nw}_{\text{P}_{35}}$ 。

使用这个技术使得超链接文本可以包含特殊文本且能够正常断行。

```
\cus_ref_label_shbox:nn {<label>} <material>
```

```
\cus_ref_label_shbox:nn
\cus_ref_target_shbox:nn
```

链接到 $\langle label \rangle$ 或 $\langle target \rangle$ 。 $\langle material \rangle$ 可以包含特殊文本，可以断行，但不允许手动断行。

另见 $\backslash\text{cus_ref_label_box:nn}_{\text{P}_{45}}$ 、 $\backslash\text{cus_ref_target_box:nn}_{\text{P}_{45}}$ 。

§ 6 struct 模块

以下简单描述 struct 模块中目录的数据结构。

```
\addcombinedlisttype {<type>} {<cbl levels>}
```

```
\addcombinedlisttype
```

若要添加新的目录类型，必须先声明 $\langle type \rangle$ 。 $\langle cbl levels \rangle$ 为这个目录类型可用的层级名称，层级名后的中括号括起的数字表示其层级 $\langle level \rangle$ ，也可使用通常的 key=val 的形式。

比如，对于标准的目录 $\backslash\text{tableofcontents}_{\text{P}_{35}}$ ，它写入的 type 为 toc ，有

```
\addcombinedlisttype{toc}
{
  part[-1],
  chapter[0],
  section[1], subsection[2], subsubsection[3],
  sub3section[4], sub4section[5],
  paragraph[4], subparagraph[5],
}
```

例 56

对于标准的 $\backslash\text{listoffigures}_{\text{P}_{35}}$ 和 $\backslash\text{listoftables}_{\text{P}_{35}}$ ，有

```
\addcombinedlisttype{lof}{ figure=1 }
\addcombinedlisttype{lot}{ table=1 }
```

例 57

原有的 `\addcontentsline` 接受三个参数，其中第一个参数为写入的文件的扩展名，在这里就是目录项的类型 $\langle type \rangle$ ，其第二个参数就是这里的 $\langle cbl level \rangle$ ，即层级名称， $\langle cbl levels \rangle$ 应包含这个参数的所有可能值；第三个参数就是 $\langle list entry \rangle$ 。新设置的值将覆盖旧有的值。

绝大多数情况下，无需手动设置它，**struct** 模块会自动设置它们。

<code>\retcbltypelevel</code> *	<code>\retcbltypelevel {$\langle type \rangle$} {$\langle cbl level \rangle$}</code>
---------------------------------	--

展开为 $\langle type \rangle$ 类型中层级名称 $\langle cbl level \rangle$ 对应的层级数。前缀 `ret` 为 `return`。

<code>\retcbltotalcounts</code> *	<code>\retcbltotalcounts {$\langle type \rangle$}</code>
-----------------------------------	---

展开为 $\langle type \rangle$ 类型的目录条目数。若 $\langle type \rangle$ 为空，则为各类型的总和。

每个类型的条目数在使用 `\enablecombinedlist`^{[P.35](#)} 时就已经确定，此后不可更改，只需常数时间即可获取（包括为空时的情况）。

每个类型的目录条目包含为如下数据：

```
\cus@type@contentsline { $\langle type \rangle$ }{ $\langle cbl count \rangle$ }{ $\langle info \rangle$ }{ $\langle level \rangle$ }
{ $\langle list entry \rangle$ }{ $\langle thepage \rangle$ }{ $\langle anchor \rangle$ }\cus@type@contentsline@
```

- `\cus@type@contentsline`，`\cus@type@contentsline@`，这两个宏标记条目的边界；
- $\langle type \rangle$ 为目录类型名；
- $\langle cbl count \rangle$ 为本条目在存储着所有目录项的那个列表中的位置；
- $\langle info \rangle$ 为一个列表，它标记着这个目录项的某些信息，第一项一般为这个目录项层级的名称，其余的项形如 $\{\langle property \rangle\}\{\langle value \rangle\}$ 。在 `\chapter`^{[P.30](#)} 中为 `chapter`，在 `figure` 中，为 `figure`；
- $\langle level \rangle$ 为这个目录项的层级，是一个整数；
- $\langle list entry \rangle$ 为这个目录项的值，通常包含着标题或 `caption`；若有编号，则为 `\numberline{ $\langle name \rangle$ }{ $\langle title \rangle$ }`， $\langle name \rangle$ 是编号， $\langle title \rangle$ 是标题；若目录项没有编号，则为 `\nonumberline{ $\langle title \rangle$ }`；若目录项的值中有 `\hspace`，则第一个未用花括号包裹的 `\hspace` 总是位于 $\langle name \rangle$ 中；
- $\langle thepage \rangle$ 为目录项所在页的 `\thepage`；
- $\langle anchor \rangle$ 为目录项原位置的锚点。仅在 `hyperref` 宏包加载时有效。可作为 `\hyperlink` 命令的第一个参数。

每个目录类型不仅分别存储在各自的列表中，还存储在一个统一的列表（以下称为 `cbl` 列表）中。在目前的版本中，存储顺序是按照宏的执行顺序而并不一定是文档实际输出顺序（例如，一个浮动体可能出现在其执行顺序之前）。

`cbl` 列表除了 `\cus@type@contentsline`、`\cus@type@contentsline@`、 $\langle cbl count \rangle$ 改为 `\cus@cbl@contentsline`、`\cus@cbl@contentsline@`、 $\langle type count \rangle$ 外，其它未改变。这里的 $\langle type count \rangle$ 为此条目在其对应类型的列表中的位置。

<code>\retcblentryname</code> *	<code>\retcblentryname {$\langle type \rangle$} {$\langle count \rangle$}</code>
---------------------------------	--

展开为存储着类型 $\langle type \rangle$ 第 $\langle count \rangle$ 项的那个宏的名称。可以使用 `\UseName`、`\@nameuse` 等获取这个目录项。也可以作为 `LATEX3` 函数的 `c` 参数，如 `\t1_show:c{\retcblentryname}{\retcbltotalcounts{}}` 在终端中显示 `cbl` 列表的最后一项。

```
\retcblentrydata {<type>} {<data>} {<count>}
```

获取 *<type>* 列表第 *<count>* 项 *<data>* 的值。*<type>* 为空, 则获取 cbl 列表中的值。

<data> 可为 type、count、info、level、entry、thepage、anchor。

```
\iteratecontents {<type>} {<inline code>}
```

使用 *<inline code>* 迭代 *<type>* 中的每一个目录项。若 *<type>* 为空, 则迭代 cbl 列表。

<inline code> 可使用 7 个参数, 分别顺序代表前述的 7 个数据值。

```
\retcbldefaultlevellistname {<type>}
```

展开为一个 clist 的名称, 这个 clist 的前 *n* 项为 *<type>* 这个类型中层级为 0 的项的索引 (即 *<type count>* 的值), 最后一项为 `\retcbltotalcounts{<type>}+1`。

当写入目录时这个 int 寄存器递增一次。

在此处, 它的值为 51, 表示此前最近一次写入的目录是目录文件中的第 51 项。

当 toc 类型的目录中添加层级为 0 的条目时这个 int 寄存器递增一次。如, 在使用不带星号的 `\chapterE30` 时将递增 1。

在此处, 它的值为 4, 表示此处的章节是第 4 个层级为 0 的章节。

```
\retcblentrydata *
```

```
\iteratecontents
```

```
\retcbldefaultlevellistname *
```

```
\CurrentCombinedListCount
```

```
\CurrentTocDefaultLevelCount
```

```
\ExplSyntaxOn
\tl_set:Nx \l_tmpa_tl { \retcbldefaultlevellistname {toc} }
\clist_item:cn { \l_tmpa_tl } { \CurrentTocDefaultLevelCount } ,~
\clist_item:cn { \l_tmpa_tl } { \CurrentTocDefaultLevelCount + 1 }
\ExplSyntaxOff
```

例 58

32, 46。表示本章节的所有目录项为 toc 类型的目录中的第 32 – 45 项。

以下代码输出本章目录。

```
\ExplSyntaxOn
\int_compare:nNnT { \retcbltotalcounts{ } } > { 0 }
{
  \tl_set:Nx \l_tmpa_tl { \retcbldefaultlevellistname {toc} }
  \int_set:Nn \l_tmpa_int % 本章开始
  { \clist_item:cn { \l_tmpa_tl } { \CurrentTocDefaultLevelCount } }
  \int_set:Nn \l_tmpb_int % 下章开始
  { \clist_item:cn { \l_tmpa_tl } { \CurrentTocDefaultLevelCount + 1 } }
  \int_step_inline:nnnn { \l_tmpa_int } { 1 } { \l_tmpb_int - 1 }
  { \tl_use:c { \retcblentryname {toc} } {#1} } }
}
\ExplSyntaxOff
```

例 59

第三章 编程接口

40

§ 1	L ^A T _E X 2 _ε 的钩子机制	41
§ 2	L ^A T _E X 2 _ε 的模板机制	42
§ 3	ltx 模块	43
§ 4	util 模块	43
3.4.1	交叉引用、超链接和书签	43
3.4.2	向前查找和收集内容	46
3.4.3	分析记号	46

3.4.4 杂项	50
§ 5 box 模块	54
3.5.1 为宽度固定和宽度可变的内容创建超链接	54
3.5.2 特殊的“水平”盒子	54
§ 6 struct 模块	55
§ 7 L ^A T _E X 2 _ε 的 mark 机制	58

`\getcbllevelrange``\getcbllevelrange {<type>} {<level>} [{<index>}] {<min>} {<max>}`

获得在 *<type>* 目录下, 包含 *<index>* 的那个具有层级 *<level>* 的块的索引范围。

例如, 此处所属的那个章节的范围是[32,45]。表示此处所在的章在 toc 类型的目录中从第 32 个开始, 到第 45 个结束。

`\cus_contents_get:nN`
`\cus_contents_type_get:nnN``\cus_contents_get:nN {<cbl count>} <tl>`
`\cus_contents_type_get:nnN {<type>} {<type count>} <tl>`

将 cbl 的第 *<cbl count>* 项 (或 *<type>* 的第 *<type count>* 项) 保存至 *<tl>* 中。如果此项不存在, 则为 `\q_no_value`。

`\cus_get_heading_level:nnN``\cus_get_heading_level:nnN {<type>} {<level name>} <tl>`

获取 *<type>* 中 *<level name>* 的 level 值, 如果不存在这样的 *<level name>*, 则为 `\q_no_value`。

关于章节标题和目录的详细用法和样例见第四章。

§ 7 L^AT_EX 2_ε 的 mark 机制

L^AT_EX 2_ε 在 2022-06-01 的发行版中引入了新的 mark 机制。本节简述这一机制, 更详细的说明请参考 `ltmarks-doc.pdf`。本说明文档的源码也使用了这个新机制。

`\NewMarkClass`
`\mark_new_class:n``\NewMarkClass {<class>}`
`\mark_new_class:n {<class>}`

声明一个新的 mark class。仅能在导言区使用。

`\InsertMark`
`\mark_insert:nn``\InsertMark {<class>} {<text>}`
`\mark_insert:nn {<class>} {<text>}`

添加 mark 到当前的垂直列中, 这个 mark 包含 *<text>* (被完全展开)。

在不能使用浮动体的地方也无法使用这个命令, 如在一个盒子中使用它们时无效。特别的, 在 multicol 等多栏环境中使用它们将无效。

`insertmark`

在执行 `\InsertMarkP.58`、`\mark_insert:nnP.58` 时, 将首先执行 `insertmarkP.58` 钩子。


```
\TopMark [<region>] {<class>}
\mark_use_last:nn {<region>} {<class>}
```

展开为 *<class>* 在 *<region>* 中相应位置的 *<text>*。

`\FirstMarkP59`、`\LastMarkP59` 分别展开为 *<region>* 的第一个、最后一个 *<text>*。

`\TopMarkP59` 展开为上一个 *<region>* 的最后一个 *<text>*。

目前, *<region>* 可选值为 page、previous-page、column、previous-column。
在多栏 (双栏) 文档中, first-column、last-column 分别代表最左列和最右列。

<region> 默认为 page。

```
\TopMark *
\FirstMark *
\LastMark *
\mark_use_top:nn *
\mark_use_first:nn *
\mark_use_last:nn *
```

```
\IfMarksEqualTF [<region>] {<class>} {<pos1>} {<pos2>} {<true>} {<false>}
\mark_if_eq:nnnnTF {<region>} {<class>} {<pos1>} {<pos2>} {<true>} {<false>}
\mark_if_eq:nnnnnnTF {<region1>} {<class1>} {<pos1>}
                      {<region2>} {<class1>} {<pos2>} {<true>} {<false>}
```

```
\IfMarksEqualTF *
\mark_if_eq:nnnnTF *
\mark_if_eq:nnnnnnTF *
```

判断两个 mark 的 *<text>* 是否完全相等 (使用 `\ifx`)。

<pos> 为 top、first、last 之一。

原有的 `\markboth`、`\markright`、`\leftmark`、`\rightmark` 仍然可用。

可用使用 `2e-left`、`2e-right`、`2e-right-nonempty class` 来获取 `\leftmark` 和 `\rightmark`。`2e-right` 与 `2e-right-nonempty` 的区别是, 后者仅在 *<rightmark>* 非空时才更新。

第四章 章节标题和目录

CuS_TE_X 重新实现了标题和目录的命令。其中标题的设置方式是 C_TE_X 风格的, 输出目录则提供了多种设置风格, 例如 `etoc` 宏包和 KOMA-Script 文档类的风格。可以在正文中任意位置使用任意次这些命令。

§ 1 title class, 标题类

标题类规定了一个标题总体上是如何显示的, 可以使用一些键值选项来微调具体的显示效果。

CuS_TE_X 预定义了 5 个标题类, 分别为: page、top、normal、free、wrap。

用户可以自己定义标题类 *<class>*。只需定义如下 6 个命令:

`\title@class@<class>` 用于显示标题的代码, 有两个参数, 第一个为标题的名称, 第二个表示是否为带星号的标题 (用 `\title@ifstar`) 判断;

`\title@classkeys@<class>` 此标题类额外的键;

`\title@classinitial@<class>` 初始值;

`\title@classhook@<class>afterdef` 钩子, 可选, 使用此标题类定义标题时, 定义结束后执行的代码;

`\title@classhook@<class>begin` 钩子, 可选, 此标题开始时要执行的代码, 一般用在 `\title@class@<class>` 中;

`\title@classhook@<class>end` 钩子, 可选, 此标题结束时要执行的代码, 一般用在 `\title@class@<class>` 中;

§ 2 输出 L^AT_EX 原始风格的目录

CuS_TE_X 接管各种目录的输出, 如标题、图表目录等。如果要输出任何一种目录, 则必须通过 `\enablecombinedlistP35` 命令启用。这个命令只能使用一次, 可

宏包 / 环境	<i>type</i>	<i>level name</i>	宏包 / 环境	<i>type</i>	<i>level name</i>
algorithm2e	loa	algocf	chemmacros	lor	reaction
hypdvips	loa	FileAttachment EmbeddedFile	musical	los lod	section
listings	lol	lol lstlistings	pdfcomment	lpc	lpcsec
poetry	lop	poem poemgroup	todonotes	tdo	todo
thmtools	loe	由 \newtheorem、\declaretheorem 定义的环境名			
figure	lof	figure	table	lot	table

表 4.1: 受支持的宏包和 figure、table 环境

以用于导言区（此时它自动移动到文档开头）和文档开头，使用它以后会读取指定的目录文件。此后方可通过 `\tableofcontentsP35` 等命令来输出目录。

默认情况下,使用 `\tableofcontentsP35`、`\listoffiguresP35`、`\listoftablesP35` 就是 L^AT_EX 默认的格式。这些命令是对 `\multicolplaincombinedlistP36` 的简单封装，即

```

\DeclareRobustCommand\tableofcontents[1][columns=1]
  {\multicolplaincombinedlist[{\#1}]{\contentsname}{toc}}
\DeclareRobustCommand\listoffigures[1][columns=1]
  {\multicolplaincombinedlist[{\#1}]{\listfigurename}{lof}}
\DeclareRobustCommand\listoftables[1][columns=1]
  {\multicolplaincombinedlist[{\#1}]{\listtablename}{lot}}

```

例 60

`\multicolplaincombinedlistP36` 则是用于输出默认的多栏目录。可以使用一个可选参数设置多栏的样式，见第 2.4.3 小节。

CuS_TE_X 的目录机制适配了许多宏包，诸如 algorithm2e、chemmacros、listings、thmtools 等有固定目录扩展名（即目录类型）的宏包，以及 float、newfloat、floatrow、tcolorbox 等可设置目录扩展名的宏包，既可以直接使用这些宏包自己的输出目录的命令，也可以使用上述的两个通用的命令来输出目录，前提是知道目录的类型。

受支持的有固定目录扩展名的宏包其扩展名和 *level name* 如表 4.1 所示，使用诸如 float 宏包创建的不在此列。暂不支持 ntheorem 宏包。

像 float 等可以自定义浮动环境的宏包，其目录类型 *type* 就是目录的扩展名，*level name* 就是所定义的浮动环境名。

对于 tcolorbox 宏包，其目录类型就是键 /tcb/new/list inside 指定的名称，*level name* 就是键 /tcb/new/list type 指定的值。

这些宏包的 *level name* 的 *level* 值都是 1，即与 `\sectionP30` 同级。可以使用 `\addcombinedlisttypeP55` 修改。

<code>\settocdepth</code>	<code>\settocdepth {<整数或层级名称>}</code>
---------------------------	---

设置 tocddepth 计数器的值。可以用来控制目录显示的层级。

§ 3 使用模板的目录

前面已经介绍了模板目录的基本使用方法。`code.name`、`code.title`、`code.leader`、`code.page`、`code.hyper` 也能够应付大部分的需要了。

但如果要使模板目录不根据目录层级名来排布, 而使用目录层级对应的整数, 由于不同的目录层级名可以对应同一个整数, 仅仅使用默认的方式将无法实现这一需求。本节介绍的内容可以帮助用户定义一个这样的模板目录。

模板目录实际上由名为 `templatecbl` 的 `socket` 和 `template` 这两种抽象结构共同作用。对于一个模板格式的目录, `socket` 和 `template` 是纵向和横向的关系。`templatecbl socket` 用于控制模板目录的整体, 而 `templatecbl object type` 用于控制目录项的内容。关于 `socket` 的作用和用法请参考 `ltsocket-doc.pdf`。

默认的按照目录层级名来排布目录项是名为 `by name` 的 `plug`, 它限定了模板目录应当如何使用和排布这些目录项。对应于该 `plug` 的是名为 `by name` 的 `template`, 它限定了目录项应当如何排布它的内容。

名为 `templatecbl` 的 `socket` 需要 5 个参数, 分别为目录类型、`templatecbl keys`、`multicolumns keys`、目录自动添加的章节标题以及目录项的范围。用户可为该 `socket` 定义 `plug`, 它用来规定如何处理目录项。

名为 `templatecbl` 的 `object type` 需要 7 个参数, 为目录条目保存的内容, 前面已经多次提及。用户可为该 `object type` 定义 `template`, 然后基于 `template` 定义实例。

如果自定义了标题命令, 每个标题命令都会定义一个与命令名相同的目录层级名, 如果要基于 `by name template` 输出目录, 则需要定义同名的实例, 如假如定义了 `\mytitle`, 且它与 `\section`_{P30} 同级, 则可以使用 `\DeclareInstanceCopy`_{P43} `{templatecbl}{mytitle}{section}` 直接复制 `section` 的定义。

§ 4 etoc 风格的目录设置方式

`etoc` 宏包提供了 `\etocsetstyle` 命令来设置目录, `CuSTeX` 提供了类似的命令。

`CuSTeX` 提供了 `\tocsetstyle`_{P39} 和 `\specifiedlot`_{P39} 来分别设置和输出目录, 它们独立于 `\tableofcontents`_{P35} 命令, 互不干扰。对于图片和表格也有类似的命令: `\lofsetstyle`_{P39}、`\specifiedlof`_{P39}、`\lotsetstyle`_{P39}、`\specifiedlot`_{P39}。设置目录和输出目录的命令都是分别对一个更加基本的命令的封装:

```
\newcommand{\tocsetstyle}{\SetSpecifiedCombinedListStyle[toc]}
\newcommand{\specifiedtoc}{\SpecifiedCombinedList[toc]}
\newcommand{\lofsetstyle}{\SetSpecifiedCombinedListStyle[lof]{figure}}
\newcommand{\specifiedlof}{\SpecifiedCombinedList[lof]}
\newcommand{\lotsetstyle}{\SetSpecifiedCombinedListStyle[lot]{table}}
\newcommand{\specifiedlot}{\SpecifiedCombinedList[lot]}
```

例 61

因此, 只需介绍 `\SetSpecifiedCombinedListStyle`_{P40} 和 `\SpecifiedCombinedList`_{P40} 这两个命令。

以下称由这两个命令制作和输出的目录为 “specified cbl”。

每个 `specified cbl` 的条目的顺序就是保存在目录文件中的顺序。每个条目都有唯一的层级, 把每个条目看成是树中的结点, 上层条目其 `level` 值更小, 下层条目其 `level` 值更大。某个条目, 设其 `level` 为 l , 连同其下 ($level > l$) 的所有条目构

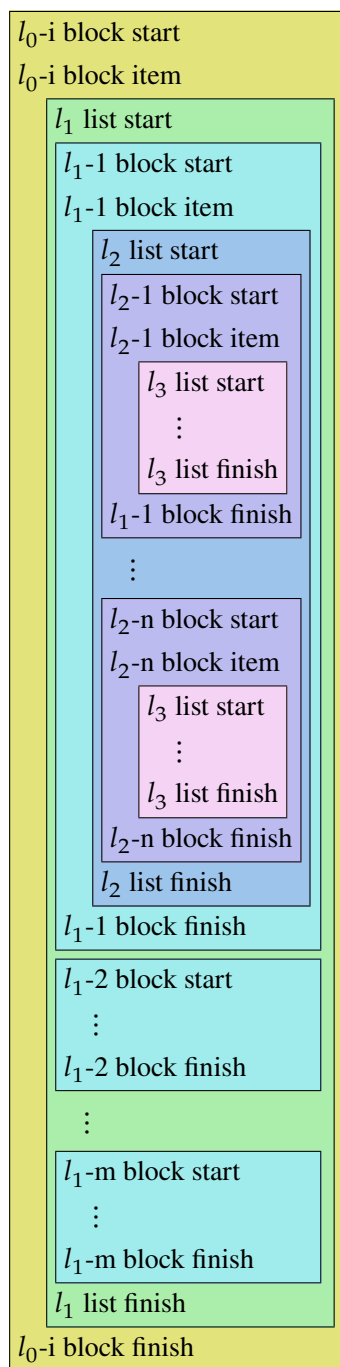


图 4.1: specified cbl 层级图

成这个条目的块。在其父条目（在它前面的最近的 $level$ 为 $l-1$ 的条目）的块中，所有 $level$ 为 l 的条目的块按其原有的顺序组成一个列表，就是 $level$ list。

`\SpecifiedCombinedListP.40` 接受一个可选参数 $\langle type \rangle$ ，即目录类型。用于输出这个目录类型的目录条目。`\SetSpecifiedCombinedListStyleP.40` 有一个可选参数和 6 个必需参数。分别为：

1. $\langle type\ list \rangle$ ，目录类型的列表；
2. $\langle level\ list \rangle$ ，层级列表，由 $level\ name$ 或数字组成的列表；
3. $\langle list\ start \rangle$ ，此层级列表开始时执行的代码；
4. $\langle block\ start \rangle$ ，此层级的块开始时执行的代码；
5. $\langle block\ item \rangle$ ，此条目执行的代码；
6. $\langle block\ finish \rangle$ ，此层级的块结束时执行的代码；
7. $\langle list\ finish \rangle$ ，此层级列表结束时执行的代码。

它们的具体位置参见图 4.1，另见例 63。

在这些参数中可以，定义了如下的命令来辅助制作目录：

`\tocthenname` 标题前的数字。例如：“第一章”、“§ 1”；可能为空，可以使用 `\tocifnamed` 命令判断；

`\tocthetitle` 标题。例如本章标题：“章节标题和目录”；

`\tocthepage` 页码。不一定是数字，可以用 `\tocifpageisnumber` 判断；

`\tocifnamed $\{\langle true \rangle\}\{\langle false \rangle\}$` ，判断该目录条目是否有编号；

`\tocifpageisnumber $\{\langle true \rangle\}\{\langle false \rangle\}$` ，判断页码是否为数字；

`\toclink $\{\langle text \rangle\}$` ，创建超链接，把 $\langle text \rangle$ 链接到文档中的对应位置；

`\toclinkbox $\{\langle context \rangle\}$` ，同上，对于非文字内容也能正确跳转；

`\tociffirst $\{\langle true \rangle\}\{\langle false \rangle\}$` ，判断当前目录条目是否是其所在的 $level\ list$ 的第一项；

`\tocifheadentry $\{\langle true \rangle\}\{\langle false \rangle\}$` ，判断当前目录条目是否是此类型目录的第一项，不考虑自动补全的；

`\tociftailentry $\{\langle true \rangle\}\{\langle false \rangle\}$` ，判断当前目录条目是否是此类型目录的最后一项，不考虑自动补全的；

`\toctheanchor` 此目录条目在文档中的位置，可作为 `\hyperlink`、`\HyperLinkP.79` 等命令的第一个参数；

`\tocthelevel` 此条目的层级；

`\toctheprevlevel` 此条目上一个条目的层级；

`\tocthenextlevel` 此条目下一个条目的层级；

`\tocthetype` 此条目所属的目录类型；

`\toctheclass` 此条目的 $level\ name$ ；

`\toctheindex` 表示该条目在所属的目录类型中是第几项；

`\tocifcomplement $\{\langle true \rangle\}\{\langle code \rangle\}$` ，specified cbl 会自动补全缺失的层级，该命令用于判断是否是自动补全的；

`\tocretinfo $\{\langle property \rangle\}$` 获得目录中的某些特殊信息；

`\toctheinfo` 目录条目中可能包含的某些特殊信息，为 `prop` 类型；

`\tocthecount` 在 cbl 中的索引；

`\tocthepreventry` 前一个条目的数据；

`\tocthenextentry` 后一个条目的数据；

`\tochilevel` 此目录中最高层级目录的 *level* 值;

`\toclolevel` 此目录中最低层级目录的 *level* 值,该值数值上不小于 `\tochilevel`。

这些命令在每个块中都是有效的,但对于自动补全的层级和 `list start`、`list finish` 中则有效的仅有 `\tocthelevel`、`\toctheclass`、`\tocifcomplement`、`\tocthetype`、`\tochilevel`、`\toclolevel`, 因为其它命令在这些内容中没有意义。

下面这个简单的例子展示了 `\SetSpecifiedCombinedListStyle`^{P40} 的强大能力 (尽管它是用 `\tocsetstyle`^{P39} 设置的)。结果如图 4.2 所示。

```
\newcommand{\ifinmiddle}[2]{\ifnum\tocthelevel=\tocthenextlevel\relax 例 62
↪ #1\else #2\fi}
% \usepackage{enumitem}
% \setlist[description]{nosep}
\tocsetstyle
{chapter,section}
{\begin{description}}
{}
{\item[\tocifnamed{\tocthename}]{\rule{1ex}{1ex}}}
\tocthetitle\quad\toclink{\tocthepage}\par}
{}
{\end{description}}
\tocsetstyle{subsection}
{\par\begin{group}\small\itshape\raggedright [ \ ]
{}
{\tocthename\enskip\tocthetitle
(\toclink{\tocthepage}) \ifinmiddle{; }{}}
{}
{ \ ] \par\endgroup\par}
\startmulticolumns[2,ragged]
\specifiedtoc
\stopmulticolumns
```

简单解释一下这个例子。首先我们定义了一个命令 `\ifinmiddle`, 用于判断是否在两个同层级的块中间, 这只需 `\tocthelevel` 和 `\tocthenextlevel` 相等即可。要输出诸如 `description` 环境效果的目录, 最简单的方法就是使用 `enumitem` 宏包, 然后设置一下间距即可。这里我们使用 `nosep` 将垂直间距设为 0pt。

然后我们在 `chapter` 列的开头和结尾分别插入 `\begin{description}` 和 `\end{description}`, 然后设置 *block item* 为 `\item`。这样, 每个 *level name* 为 `chapter` 的目录条目都作为 `description` 环境的一项。然后把超链接设置到页码上, 这样点击页码就能跳转到文档的对应位置。对 `section` 做同样的事情。

最后, 我们设置 `subsection`。首先, 在 `subsection` 列的开始处我们修改它的字体, 为了不破坏其它地方的字体, 我们把它放到一个组中。此列的开头和结尾分别显示 “**[**” “**]**”。之后, 我们照常设置标题的页码, 并使用括号括住带有超链接的页码。最后, 用到了先前定义的 `\ifinmiddle`, 如果在中间, 则插入一个分号分隔。

想要双栏并排显示只需使用 `\startmulticolumns`^{P23}。

下面这个例子展示了本文目录在 CuS_TE_X 目录机制下的输出顺序。

```
\startmulticolumns[cols=4,ragged,column-sep=10pt] 例 63
\newcommand\showthelevel{$1_{\tocthelevel}$ }
\newcommand\showhbarl{\Replicate{2*(\tocthelevel-\tochilevel)}{|}}
\newcommand\showhbarb{\Replicate{2*(\tocthelevel-\tochilevel)+1}{|}}
\tocsetstyle{chapter,section,subsection,0,1,2}
```

■ 总目录	i	§ 2 tcb 库	72
第一章 概述	1	【5.2.1 multicolumns/framed=tcbox (72)】	
第二章 文档接口	1	§ 3 logo 库	72
§ 1 ltx 模块	2	§ 4 doc 库	72
【2.1.1 参数处理器, <i>Argument processors</i> (5)】		§ 5 bnf 库	76
§ 2 util 模块	7	§ 6 ref 库	79
§ 3 页面布局, layout 模块	9	§ 7 box 库	80
【2.3.1 页面尺寸 (9); 2.3.2 主体尺寸 (10);		【5.7.1 paracol 环境 (80); 5.7.2	
2.3.3 边距 (13); 2.3.4 原有的变量 (13); 2.3.5		multicolumns/framed=lfbox (83); 5.7.3	
页眉页脚 (14); 2.3.6 杂项 (15); 2.3.7 设置页		\fpabox _{P83} 和 \fvarbox _{P83} , 可设置外框的命令	
眉页脚 (15)】		(83)】	
§ 4 盒子和填充, box 模块	17	§ 8 math 库	83
【2.4.1 Framed (17); 2.4.2 Filler (19); 2.4.3 多		§ 9 counter 库	83
栏文字 (23); 2.4.4 额外增加文字的宽度 (26);		§ 10 pdf 库	84
2.4.5 旋转的盒子 (26)】		第六章 可单独加载的宏包	85
§ 5 背景, bgfg 模块	27	§ 1 collectn	85
§ 6 索引, index 模块	28	§ 2 lt3ekeys	90
§ 7 文档结构, struct 模块	29	【6.2.1 定义键 (90); 6.2.2 设置键 (90); 6.2.3	
【2.7.1 初始化设置 (30); 2.7.2 编号 (31); 2.7.3		lt3ekeys-elkernel (90); 6.2.4 定义命令——	
格式 (32); 2.7.4 间距和缩进 (32); 2.7.5 浮动		lt3ekeyscmd (90); 6.2.5 定义命令扩展——	
体 (33); 2.7.6 杂项 (33); 2.7.7 目录 (35)】		lt3ekeysext (95)】	
§ 8 buffer 模块	40	■ TODO	102
第三章 编程接口	40	■ 索引	103
§ 1 L ^A T _E X 2 _ε 的钩子机制	41	■ 代码索引	103
§ 2 L ^A T _E X 2 _ε 的模板机制	42	■ List of Hackings	119
§ 3 ltx 模块	43	■ cus.module.ltx.tex	119
§ 4 util 模块	43	■ cus.module.util.tex	119
【3.4.1 交叉引用、超链接和书签 (43); 3.4.2 向		■ cus.module.algo.tex	119
前查找和收集内容 (46); 3.4.3 分析记号 (46);		■ cus.module.layout.tex	119
3.4.4 杂项 (50)】		■ cus.module.box.tex	120
§ 5 box 模块	54	■ cus.module.bgfg.tex	120
【3.5.1 为宽度固定和宽度可变的内容创建超链接		■ cus.module.index.tex	120
(54); 3.5.2 特殊的“水平”盒子 (54)】		■ cus.module.struct.tex	120
§ 6 struct 模块	55	■ cus.library.box.tex	121
§ 7 L ^A T _E X 2 _ε 的 mark 机制	58	■ cus.library.math.tex	121
第四章 章节标题和目录	59	■ cus.library.counter.tex	121
§ 1 title class, 标题类	59	■ cus.library.ref.tex	121
§ 2 输出 L ^A T _E X 原始风格的目录	59	■ cus.library.pgfg.tex	121
§ 3 使用模板的目录	61	■ cus.library.tcb.tex	122
§ 4 etoc 风格的目录设置方式	61	■ cus.library.pdf.tex	122
§ 5 目录的内部处理方式	69	■ lt3ekeys、lt3ekeyscmd 和 lt3ekeysext	122
第五章 库的文档接口	69	■ lt3ekeys-elkernel	122
§ 1 pgf 库	69	■ lt3ekeys-collectn	122
【5.1.1 文字渐变 (70); 5.1.2 在背景和前景中使用		■ updatemarks	122
TikZ 绘制 (71)】			

图 4.2: 例 62 的结果

[illegible]

[illegible]

下例展示了为目录添加盒子和边距的方法：

```

\startmulticolumns[2,ragged]

\colorlet{tocgreen}{green!70!black}

\newcommand{\tochyperpage}{\toclink{\tocthepage}}

\hypersetup{hidelinks}

\makeatletter

\tocsetstyle {chapter,0}
{
  {\noindent}
  {\fparbox{\linewidth}[border-color=tocgreen, background-color=tocgreen,
padding={0pt,\fboxsep}]
  {\bfseries\large\raggedright \hangindent4\ccwd \hangafter1
  \color{white}%
  \strut \tocifnamed{\tocthenname\quad\kern-.3em}{\tocthetitle
  \breakablefiller[space]\tochyperpage \strut\par}\par }
  {\smallskip}
  {}
}

\tocsetstyle {section,1}
{
  {\smallskip
  \begin{list}{}{\leftmargin3\ccwd \labelsep\z@
  \itemindent-\ccwd \listparindent\itemindent
  \topsep\z@ \partopsep\z@ \itemsep\z@ \parsep\z@ \parskip\z@}
  {\item \begingroup\color{tocgreen}\bfseries}
  {\tocifnamed{\tocthenname\quad}{\tocthetitle\breakablefiller[space]%
  \makebox[1.5em][r]{\tochyperpage\;}\par }

```

```

{\endgroup}
{\end{list}}
\tocsetstyle {subsection,2}
{
  {\begingroup\color{black}\bfseries}
  {\tocifnamed{\tocthenname\quad}{\tocthetitle\breakablefiller[dotted]}%
    \makebox[1.5em][r]{\tochyperpage\;} \par }
  {\endgroup}
  {}
\makeatother

\specifiedtoc

\stopmulticolumns

```

总目录	i	§ 8 buffer 模块	40
第一章 概述	1	第三章 编程接口	40
第二章 文档接口	1	§ 1 L ^A T _E X 2 _ε 的钩子机制	41
§ 1 ltx 模块	2	§ 2 L ^A T _E X 2 _ε 的模板机制	42
2.1.1 参数处理器, Argument processors..	5	§ 3 ltx 模块	43
§ 2 util 模块	7	§ 4 util 模块	43
§ 3 页面布局, layout 模块	9	3.4.1 交叉引用、超链接和书签	43
2.3.1 页面尺寸	9	3.4.2 向前查找和收集内容	46
2.3.2 主体尺寸	10	3.4.3 分析记号	46
2.3.3 边距	13	3.4.4 杂项	50
2.3.4 原有的变量	13	§ 5 box 模块	54
2.3.5 页眉页脚	14	3.5.1 为宽度固定和宽度可变的内容创建超链接	54
2.3.6 杂项	15	3.5.2 特殊的“水平”盒子	54
2.3.7 设置页眉页脚	15	§ 6 struct 模块	55
§ 4 盒子和填充, box 模块	17	§ 7 L ^A T _E X 2 _ε 的 mark 机制	58
2.4.1 Framed	17	第四章 章节标题和目录	59
2.4.2 Filler	19	§ 1 title class, 标题类	59
2.4.3 多栏文字	23	§ 2 输出 L ^A T _E X 原始风格的目录	59
2.4.4 额外增加文字的宽度	26	§ 3 使用模板的目录	61
2.4.5 旋转的盒子	26	§ 4 etoc 风格的目录设置方式	61
§ 5 背景, bgfg 模块	27	§ 5 目录的内部处理方式	69
§ 6 索引, index 模块	28	第五章 库的文档接口	69
§ 7 文档结构, struct 模块	29	§ 1 pgf 库	69
2.7.1 初始化设置	30	5.1.1 文字渐变	70
2.7.2 编号	31	5.1.2 在背景和前景中使用 TikZ 绘制	71
2.7.3 格式	32	§ 2 tcb 库	72
2.7.4 间距和缩进	32	5.2.1 multicolumns/framed=tcbbox	72
2.7.5 浮动体	33	§ 3 logo 库	72
2.7.6 杂项	33	§ 4 doc 库	72
2.7.7 目录	35		

§ 5	bnf 库	76	代码索引	103
§ 6	ref 库	79	List of Hackings	119
§ 7	box 库	80	cus.module.ltx.tex	119
5.7.1	paracol 环境	80	cus.module.util.tex	119
5.7.2	multicolumns/framed=lfbox	83	cus.module.algo.tex	119
5.7.3	\fparbox _{P.83} 和 \fvarbox _{P.83} , 可设置外框的命令	83	cus.module.layout.tex	119
§ 8	math 库	83	cus.module.box.tex	120
§ 9	counter 库	83	cus.module.bgfg.tex	120
§ 10	pdf 库	84	cus.module.index.tex	120
第六章 可单独加载的宏包			cus.module.struct.tex	120
§ 1	collectn	85	cus.library.box.tex	121
§ 2	lt3ekeys	90	cus.library.math.tex	121
6.2.1	定义键	90	cus.library.counter.tex	121
6.2.2	设置键	90	cus.library.ref.tex	121
6.2.3	lt3ekeys-elkernel	90	cus.library.pgf.tex	121
6.2.4	定义命令——lt3ekeyscmd	90	cus.library.tcb.tex	122
6.2.5	定义命令扩展——lt3ekeysext	95	cus.library.pdf.tex	122
TODO			lt3ekeys、lt3ekeyscmd 和 lt3ekeysext	122
索引			lt3ekeys-elkernel	122
			lt3ekeys-collectn	122
			updatemarks	122

还可使用 `\LocalSpecifiedCombinedListP.40` 来输出局部目录。`\localspecifiedtocP.39` 是一个特例，用来输出局部章节的目录。这个命令除了有一个用来设置目录类型的可选参数外，还支持修改局部目录的层级和条目的位置。

例如，此处处在在 `\sectionP.30` 中，输出的局部目录为此 `\sectionP.30` 小节的目录。但可以设置 `\localspecifiedtocP.39(chapter)` 来输出本章节的目录。还可以通过修改条目的位置来修改输出的章节。例如 `\localspecifiedtocP.39(chapter,\?-4)` 则是输出的本节目录条目往前数第 4 个条目所在章的目录。这里 `\?` 就是 `\CurrentCombinedListCountP.57`。还有一个变量 `\$`，它表示此刻在该类型的目录中的项数。

关于局部章节目录，另见例 59。

下例输出本章目录。

<pre> \begin{group} \tocsetstyle {chapter,0} { {\startmulticolumns[ragged,outer-sep=0pt, heading={\centering\Large\bfseries\tocthetitle\par}} } {\stopmulticolumns} \tocsetstyle {section,subsection} { \begin{description} } {\tocifcomplement{\item[\rule{1ex}{1ex}]{\rule{1ex}{1ex}}\par} {\item[\tocifnamed{\tocthename}]{\rule{1ex}{1ex}} \tocthetitle\quad\toclink{\tocthepage}\par}} </pre>	例 65
--	-------------

```
{\end{description}}
\localspecifiedtoc{chapter}
\endgroup
```

章节标题和目录

§ 1 title class, 标题类	59	§ 4 etoc 风格的目录设置方式	61
§ 2 输出 L ^A T _E X 原始风格的目录	59	§ 5 目录的内部处理方式	69
§ 3 使用模板的目录	61		

§ 5 目录的内部处理方式

在 CuSTeX 中, 如果要输出某类目录, 则必须先通过 `\addcombinedlisttypeP.55` 注册它, 这个命令接受两个参数, 第一个参数为 *type*, 表示添加的这个目录类型。对于标题目录, 则为 *toc*, 对于图片、表格目录则分别为 *lof*、*lot*, 它就是 L^AT_EX 标准目录输出方式里的 *ext*, 即输出目录的文件扩展名。由于 CuSTeX 把所有类型的目录都写入到同一个文件中, 因此, *type* 用于唯一区分不同的目录类型。

同一个目录中, 可以有不同的层级, 同一层级也可以有所区分。

在 CuSTeX 中, 用于区分不同层级的就是 *level* 变量, 它是一个整数, 数值小的, 层级越高。例如, 对于 *toc* 目录, 在默认情况下, `\partP.30` 为 -1, `\chapterP.30` 为 0, `\sectionP.30` 为 1, 依此类推。对于图表目录, 默认只有一个层级, 为 1。

对于同一个层级, 用以区分的就是不同的层级名 *level name*。写入目录时就是根据层级名写入的, 目录项首先根据 *level name* 进行分类, 然后再根据这些 *level name* 所属的 *level* 分类。每个类型的目录项的类别必须属于这些此类型的 *level name* 之一。因此每个类型的目录其 *level name* 必须完整。

例如, 默认情况下, *toc* 类型的目录条目都是由 `\partP.30`、`\chapterP.30`、`\sectionP.30`、`\subsectionP.30`、`\subsubsectionP.30`、`\paragraphP.30`、`\subparagraphP.30` 这些命令之一写入的, 并且它们都是可区分的。因此 *toc* 的 *level name* 至少有 7 个, 分别代表这 7 种之一。

在标准文档类下, L^AT_EX 通过 `\l@<name>` 来唯一区分这些 *level name*。但在 CuSTeX 宏集中, 每个 *level name* 只需在其所属的那个目录类型中唯一, 不同目录类型可以有相同的 *level name*。

例如, 对于图片类型的目录 *lof*, 其 *level name* 为 *figure*, 但你尽可以把它设置成 *section*, 但写入目录文件中的目录条目也必须随之修改。

尽管看起来很复杂, 但实际上以上这些工作在绝大多数情况下 CuSTeX 都会自动完成它们, 无需手动设置它。

第五章 库的文档接口

§ 1 pgf 库

pgf 库使用 pgf 宏集的功能, 定义了一些命令。

pgf 库不会自动加载 pgf 和 tikz 宏包, 需要用户自行加载。

```
\pgfdeclareimagep
\pgfimagep
```

```
\pgfdeclareimagep [image options] {image name} {filename}
\pgfimagep [image options] {filename}
```

它们的功能和 `\pgfdeclareimage`、`\pgfimage` 相同。

它们会自动查找 `\setgraphicspath`_{P.4} 设置的路径，且可以自动补全文件扩展名。

5.1.1 文字渐变

需要加载 `tikz` 宏包。

```
\shadetext
\shadetextbox
\shadecontent
\shadecontentbox
```

```
\shadetext {shade options} {text}
\shadetextbox {shade options} {material}
\shadecontent [node options] {shade options} {content}
\shadecontentbox [node options] {shade options} {material}
```

这几个命令用于为文字增加渐变效果。`\shadecontent`_{P.70} 和 `\shadecontentbox`_{P.70} 需要阅读器支持才能正确显示。

`\shadetextbox`_{P.70} 和 `\shadecontentbox`_{P.70} 的 `<material>` 的参数可以包含特殊文本，如 `verbatim` 和分段。

`\shadetext`_{P.70} 和 `\shadetextbox`_{P.70} 中，仅文字才会显示渐变效果，其它内容如方框不会显示。`\shadecontent`_{P.70} 和 `\shadecontentbox`_{P.70} 中，可见的内容都会显示渐变效果。

它们的内容不能断行，正如 `\fcolorbox` 和 `\tcbox` 那样。

例 66

```
\shadetext{left color=red,right color=blue}{\bfseries shaded}
\shadetextbox{left color=red,right color=blue}{\verb|\relax|}

\shadecontent{left color=red,right color=blue}{\bfseries shaded}
\shadecontentbox{left color=red,right color=blue}{\verb|\relax|}

\shadetext{left color=red,right color=blue}{\fbox{shaded}} % 方框不会显示
\shadecontent{left color=red,right color=blue}{\fbox{shaded}}

% \usetikzlibrary{shadings}
\shadecontent{shading=color wheel}{%
  \tikz\node[inner sep=0pt,outer sep=0pt,circle,draw,line width=1pt]
  {\parbox{4em}{\linespread{1}\selectfont \centering
    使用\\色轮作为\\渐变}};}
```

shaded `\relax`

shaded |



理想效果见图 5.1 和图 5.2。



图 5.1: 透明背景下的理想效果



图 5.2: 白色背景下的理想效果

```
tangent=(pos)
tangent=(name) at(pos)
tangent=(name) at(pos) and ...
at tangent=(name)
```

```
/tikz/tangent
/tikz/at tangent
```

`/tikz/tangent`_{P71} 用于记录一条路径在 $\langle pos \rangle$ 的切点坐标及切线方向，切点坐标为 $(\text{tangent point } \langle name \rangle)$ 。

`/tikz/at tangent`_{P71} 用于把该路径或 `scope` 的坐标系变换至指定的位置和方向。

本功能需要自行加载名为 `decorations.markings` 的 pgf 库。 $\langle pos \rangle$ 和坐标系的方向与 `/tikz/decoration/mark` 的可用值 `at position $\langle pos \rangle$ with $\langle code \rangle$` 一致。

5.1.2 在背景和前景中使用 TikZ 绘制

需要加载 `tikz` 宏包。

本库为 `\background`_{P27} 和 `\foreground`_{P27} 命令的 $\langle \text{位置} \rangle$ 参数增加了一个可用值 `tikz`。可以在 $\langle code \rangle$ 中使用 TikZ 绘图代码，坐标原点为纸张左下角。并预先定义了几个 `node`，它们只能在 `\foreground`_{P27} 和 `\background`_{P27} 命令中使用：

`page` 该 `node` 占满整个纸张，四角为纸张四角；

`layout` 该 `node` 占满整个 `layout`，四角为 `layout` 四角；

`text` 该 `node` 占满整个正文区域，四角为正文的四角；

`margin` 该 `node` 占满整个旁注区域，四角为旁注四角；

`header` 该 `node` 占满整个页眉，包括左右的偏移量，四角为页眉四角；

`footer` 该 `node` 占满整个页脚，包括左右的偏移量，四角为页脚四角。

相比于 `tikzpagenodes` 提供的 `node`，它们只需编译一次，且页眉页脚包含偏移量。

```
\background(tikz){
  \draw[thick,help lines] (layout.south west) grid (layout.north east);
  \draw[thick,blue] (layout.center) circle (2cm);
  \draw[thick,red] (page.north west)--(page.south east);
  \draw[thick,blue] (layout.north east)--(layout.south west);
  \draw[thick,cyan] (text.south west)rectangle(text.north east);
  \draw[thick,cyan] (margin.south west)rectangle(margin.north east);
  \draw[thick,cyan] (header.south west)rectangle(header.north east);
  \draw[thick,cyan] (footer.south west)rectangle(footer.north east);
}
```

例 67

如本页所示。

§ 2 tcb 库

tcb 库使用 tcolorbox 宏包的功能，定制了一些其它功能。用户需自行加载 tcolorbox 宏包。

5.2.1 multicolumns/framed=tcbbox

本库提供为多栏文字的外框提供了 tcbbox 可选值。表示用 tcolorbox 宏包的 \tcbbox 作为盒子外框。[multicolumns/framed-options](#)_{P24} 可以使用 tcolorbox 宏包提供的选项。

§ 3 logo 库

本库使用 hologo 宏包定义了一些 logo 命令。

\ApLaTeX	Ap _L A _T E _X	\ApTeX	Ap _T E _X	\BibTeX	Bib _T E _X	\ConTeXt	Con _T E _X t
\CusLaTeX	Cu _S L _A T _E X	\CUSLATEX	Cu _S L _A T _E X	\CusTeX	Cu _S T _E X	\CUSTEX	Cu _S T _E X
\dvipdfmx	DVIPDFM _x	\eTeX	ε- _T E _X	\iniTeX	ini _T E _X	\LaTeX	L _A T _E X
\LATEX	L _A T _E X	\LaTeXe	L _A T _E X 2 _ε	\LATEXe	L _A T _E X 2 _ε	\LaTeXiii	L _A T _E X3
\LaTeXTeX	(L _A) _T E _X	\LuaHBTeX	LuaHB _T E _X	\LuaLaTeX	LuaL _A T _E X	\LuaMetaTeX	LuaMeta _T E _X
\LuaTeX	Lua _T E _X	\LyX	L _y X	\METAFONT	META _F ONT	\MetaFun	Meta _F un
\METAPOST	META _P OST	\MiKTeX	MiK _T E _X	\pdfLaTeX	pdf _L A _T E _X	\PDFLaTeX	pdf _L A _T E _X
\pdfTeX	pdf _T E _X	\plainTeX	plain _T E _X	\pTeX	p _T E _X	\pupTeX	(u)p _T E _X
\TeX	T _E X	\TEX	T _E X	\TeXLive	T _E X Live	\TikZ	Ti _k Z
\upLaTeX	up _L A _T E _X	\upTeX	up _T E _X	\XeLaTeX	X _e L _A T _E X	\XeTeX	X _e T _E X

它们可以直接作为章节命令的参数，以及 \MakeUppercase、\MakeLowercase、\MakeTitlecase 的参数。

§ 4 doc 库

doc 库用于支持排版说明文档。本库移植修改自 l3doc 和 ctxdoc。

当加载本库时，将创建两个索引，名为 docusage 和 docchange，分别记录代码和版本历史。使用 \PrintUsages、\PrintChanges 分别输出代码索引和版本历史。可使用 no-index-file 库选项来阻止创建它们。

本库的详细用法可参考本说明文档的源码。

本库提供 \cs_{P72}、\cmd_{P72}、\tn_{P72} 来排版宏。

\cs	\cs [<i><cmd key-val></i>] { <i><macro name></i> }
\cmd	\cmd [<i><cmd key-val></i>] { <i><macro></i> }
\tn	\tn [<i><cmd key-val></i>] { <i><tex macro name></i> }
\key	\key [<i><cmd key-val></i>] { <i><key name></i> }
\cus@doc@cs@format	
\cus@doc@cmd@format	
\cus@doc@tn@format	
\cus@doc@key@format	

排版宏。 \tn_{P72} 专用于排版 T_EX 和 L_AT_EX 2_ε 的宏。

\cus@doc@cs@format_{P72} 等命令则是修改显示的格式，最多可带有一个参数。这些命令中可以修改链接显示的颜色和字体。

<code>index</code>	<code>= {⟨index entry⟩}</code>	自动设置值	doc/cmd/index
<code>module</code>	<code>= {⟨module name⟩}</code>	自动设置值	doc/cmd/module
<code>no-index</code>	<code>= ⟨true false⟩</code>	重设为: false	doc/cmd/no-index
<code>do-index</code>	<code>= ⟨true false⟩</code>	自动设置值	doc/cmd/space
<code>space</code>	<code>= ⟨true false⟩</code>	初始值: false	doc/cmd/keyval
<code>keyval</code>	<code>= ⟨true false⟩</code>	自动设置值	doc/cmd/hyper
<code>hyper</code>	<code>= ⟨true false raw⟩</code>	初始值: true	doc/cmd/target
<code>no-hyper</code>		不可设置值	doc/cmd/type
<code>target</code>	<code>= {⟨target⟩}</code>	自动设置值	
<code>target*</code>	<code>= {⟨target⟩}</code>		
<code>type</code>	<code>= {⟨label type⟩}</code>		

`\csP.72`, `\cmdP.72` 和 `\tnP.72` 可用的键值选项。

`no-index` 控制是否写入索引文件。`do-index` 与 `no-index` 相反。

当在 `functionP.74`, `keyvalP.74` 和 `syntaxP.74` 环境中时, 不写入索引文件。

`space` 控制是否将空格替换为 `\textvisiblespace`。

`keyval` 控制是否为键值选项。

`hyper` 控制是否自动链接到该命令的说明处, 仅当该命令的说明存在时有效。

`target` 设置链接的位置。默认情况下, 还会对 `⟨target⟩` 进行一定的处理, 如果已经可以确定无需再对 `⟨target⟩` 进行处理, 可以设置 `hyper=raw`。`target*=` 是 `hyper=raw`, `target=` 的简写。

`module` 设置的是在索引中归属的类别, 对于 L^AT_EX₃ 的命令, 可以自动检测其 `module`, 若要把某些命令归类到特定的 `module` 中, 则需要设置此键。

`type` 用于设置自动添加的 `label` 的类别, 命令默认为 `function`, 键名默认为 `keyval`, 环境为 `environment`, 如若修改了此值, 在引用时也必须修改。

`module` 有几个特殊的值 `hookpoint`、`colorname`, 分别用于标记此命令为钩子、颜色名称。使用这几个值时最好也将 `type` 也改为相同的值。

<code>hyphen-opacity</code>	<code>= {⟨0--1 之间的数⟩}</code>
-----------------------------	------------------------------

[doc/cmd/hyphen-opacity](#)

设置断行字符的透明度。

<code>break-at-any</code>	<code>= ⟨true false⟩</code>
---------------------------	-----------------------------

初始值: **false**

[doc/cmd/break-at-any](#)

设置是否在任何位置都可断行。设置此选项为真时会增加编译时间, 尽可能仅在必要时使用。

<u>meta</u>	排版宏的参数。
<u>veta</u>	<code>\cus@doc@meta@format_{P.74}</code> 等命令用于修改显示的格式。
<u>marg</u>	<code>\Arg_{P.74}</code> 相当于 <code>\marg_{P.74}</code> 。
<u>Arg</u>	<code>\file_{P.74}</code> 和 <code>\docfile_{P.74}</code> 相当于 <code>\nolinkurl</code> 。不提供修改格式的接口。
<u>oarg</u>	注意 <code>\marg_{P.74}</code> 、 <code>\oarg_{P.74}</code> 、 <code>\parg_{P.74}</code> 都包含 <code>\meta_{P.74}</code> 。
<u>parg</u>	
<u>pkg</u>	
<u>env</u>	
<u>cls</u>	
<u>opt</u>	
<u>file</u>	
<u>docfile</u>	
<u>\cus@doc@meta@format</u>	
<u>\cus@doc@veta@format</u>	
<u>\cus@doc@marg@format</u>	
<u>\cus@doc@oarg@format</u>	
<u>\cus@doc@parg@format</u>	
<u>\cus@doc@pkg@format</u>	
<u>\cus@doc@env@format</u>	
<u>\cus@doc@cls@format</u>	
<u>\cus@doc@opt@format</u>	

<u>function</u>	<pre>\begin{function} [<i>{function key-val}</i>] <i>{functions clist}</i> ... \end{function}</pre>
-----------------	---

显示函数说明。*{functions clist}* 可以是宏或者环境名，不可包含多余的空格以及注释符。

<u>keyval</u>	<pre>\begin{keyval} [<i>{function key-val}</i>] <i>{keys clist}</i> ... \end{keyval}</pre>
---------------	--

显示键值选项的说明。*{keys clist}* 为键列表，不可包含多余的空格以及注释符。

<u>syntax</u>	<pre>\begin{syntax} ... \end{syntax}</pre>
---------------	--

输出使用方法。

本环境中每个输入行都为输出行（一个段落），除每行首尾的空格被移除外，所有的空格都被保留下来；此外，可使用 `~` 输出一个空格的宽度。

本环境中可以使用几个特殊的字符（字符对），它们是语法糖：

- `<...>` — 在文本环境时这相当于 `\meta{...}`，数学环境时仍然为小于、大于号；但有几个例外：
- `<&...>` — 当文本环境中 `<` 紧跟 `&` 时，`...` 被视为可选值；
- `<{...}>` — 当文本环境中 `<...>` 中的内容为一个正确嵌套的组时，它被视为 `\marg{...}`；
- `&` — 其后的值被认为是初始值，每行最多应仅使用一次，与之等价的写法是：`\initialval ...`（无需花括号）；但有几个例外：
- `&*` — 当 `&` 紧跟 `*` 时，相当于 `\repinitval`，可自行设置文字；
- `&&` — 当 `&` 紧跟 `&` 时，相当于 `\forbiddenval`，表示禁止设置值；
- `&#` — 当 `&` 紧跟 `#` 时，相当于 `\automaticval`，表示如未给出将自动设置值；
- `&~` — 当 `&` 紧跟 `~` 时，相当于 `\initemptyval`，表示初始为空；

- `&!` — 当 `&` 紧跟 `!` 时, 相当于 `\resetval`, 表示在对应命令或环境中其值均被重设;
- `|` — 相当于 `|` (`\orbar`), 一般用于分隔不同的可选值;
- `(...)` — 这中间的值被认为是默认值, 以粗体显示, 与之等价的写法是: `\defaultval{...}`。

T_EXhackers note: # 在本环境中的类别码被设置为 12 (other), `<|` (`~&` 的类别码为 13 (active))。

`function`_{P74}, `keyval`_{P74} 和 `syntax`_{P74} 环境均可使用 `\V` 命令, 它和 `\Verbatimize`_{P9} 一样, 但以当前字体显示。

EXP	不可设置值	doc/function/EXP
rEXP	不可设置值	doc/function/rEXP

EXP 将函数标记为完全可展的 (fully expandable functions), 可同时用作 `x`、`e`、`f` 类型的参数。如 `\string`、`\cs_to_str:N`。使用 `★` 标记。

rEXP 将函数标记为受限可展的 (restricted expandable functions), 这些函数是完全可展的, 但不能在 `f` 类型的参数中完全展开 (cannot be fully expanded)。如 `\seq_map_function:NN`。使用 `☆` 标记。

TF	不可设置值	doc/function/TF
pTF	不可设置值	doc/function/pTF
noTF	不可设置值	doc/function/noTF

标记函数为是带有真假值参数的函数。

TF 将函数标记为带有真假参数的函数, 如 `\tl_if_eq:nnTF`。pTF 在 TF 的基础上, 还将函数标记为带有可用于 `\if_predicate:w` 的函数。noTF 在 TF 的基础上, 还将函数标记为不带真假参数的函数, 如 `\prop_get:NnN`。

added = {<年>-<月>-<日>} 或 {<年>/<月>/<日>}	doc/function/added
updated = {<年>-<月>-<日>} 或 {<年>/<月>/<日>}	doc/function/updated

此函数是何时添加的或最近一次修改在何时。

label = {<label list>}	doc/function/label
label* = {<label list>}	doc/function/label*
no-label	不可设置值
	doc/function/no-label

设置 `\label`。label 不会设置默认的 label, label* 会设置默认的 label。

verb	不可设置值	doc/function/verb
------	-------	-------------------

将整个 *(functions clist)* 或 *(keys clist)* 看作是一个函数或键。

module = {<module name>}	doc/function/module
--------------------------	---------------------

设置当前函数所在的模块。

type = {<类型>}	doc/function/type
---------------	-------------------

设置当前的类型, 如 `function`、`environment`、`keyval`。

path = {<key path>}	doc/function/path
---------------------	-------------------

设置键值参数的键路径。

```
doc/function/frame
doc/function/frame+
```

```
frame = {\frame key-val}
frame+ = {\frame key-val}
```

设置外部方框盒子的选项。

```
texnote
```

```
\begin{texnote}
...
\end{texnote}
```

```
\csref
\csreflist
\envref
\envreflist
\keyref
\keyreflist
\cus@doc@csref@format
\cus@doc@envref@format
\cus@doc@keyref@format
```

```
\csref      [{<type>}] {\cs name}
\csref      [{<type>}] (<cs label>) {\cs name}
\csreflist  [{<type>}] {\cs name list}
\envref     [{<type>}] {\env name}
\envref     [{<type>}] (<cs label>) {\env name}
\envreflist [{<type>}] {\env name list}
\keyref     [{<key path>}] {\key name}
\keyref     [{<key path>}] (<key label>) {\key name}
\keyreflist [{<key path>}] {\key name list}
```

引用命令，环境或键。对于列表的引用，可以通过 `\cus@doc@refrange` 修改分隔字符。

`\cus@doc@csref@format`^{P.76} 等三个命令则是修改它们显示的格式，最多可带有一个参数。

`<type>` 为 `doc/cmd/type`^{P.73} 设置的值。

```
\cus@doc@ttfont
\cus@doc@itfont
```

这两个命令分别用于设置 **doc** 库中使用的等宽字体和斜体。

```
cus/color/doc cs
cus/color/doc env
cus/color/doc key
```

它们是颜色名，分别用于设置 `\csref`^{P.76}、`\envref`^{P.76} 和 `\keyref`^{P.76} 命令中链接的颜色。可以使用 `\colorlet` 等命令修改。

```
\g_cus_doc_type_alias_prop
\g_cus_doc_module_alias_prop
```

这两个 `prop` 分别用于设置 `type` 和 `module` 的实际值。

§ 5 bnf 库

bnf 库用于排版基于 Backus-Naur Form (BNF 范式) 的文法。

```
\begin{latexbnf}[<texbnf key-val>]
  <(non-terminal)> : <(non-terminal)>
  <(non-terminal)> : "<(terminal)>"
  <(non-terminal)> : <(non-terminal)> | <(non-terminal)>
  ...
\end{latexbnf}
```

 latexbnf

BNF 范式排版环境。

可使用 `:=` 代替 `:`。

当 `<` 在行首时，被解释为定义一个新的句法。

在此环境中，`_`、`^` 相当于 `\loP3`、`\hiP3`，可以直接在文本中使用，分别表示上下标。

排版时，既可使用这种字符标记的形式，也可使用下述的命令形式。混合使用它们也是可被接受的。

这些字符标记中的文字被正常处理。

连续使用两次：可输出 “:”，连续使用两次 `|` 可输出 “|”。

这些标记字符在数学模式中表示它们原本的含义。

此环境中空行被忽略了，若要显示空行，可以在此行使用 `\null` 或使用一个空盒子：`\mbox{}`。

本环境中还可使用 `\V`，它相当于 `\VerbatimizeP9`。

```
\BNFN {<(non-terminal)>}
\BNFN * {<(non-terminal)>}
\BNFN + <(token)> {<(non-terminal)>} <(token)>
\BNFT {<(terminal)>}
\BNFT * {<(terminal)>}
\BNFT + <(token)> {<(terminal)>} <(token)>
```

 \BNFItem
 \BNFN
 \BNFI
 \BNFO
 \BNFT

`\BNFItemP77` 用于标记一个句法 (syntax) 的开始。

`\BNFNP77` 排版非终结符。`\BNFTP77` 排版终结符。

`\BNFIP77` 表示它之前的内容被定义为它之后的内容。

`\BNFOP77` 表示 “或者”。

除 `\BNFItemP77` 外，上述命令均可在正文环境中使用。

在 `latexbnfP77` 环境中，可使用 `\is` 代替 `\BNFIP77`，`\alt` 代替 `\BNFOP77`，它们会在两侧加上空白。

本库还支持给非终结符加上超链接。

当加载了 `hyperref` 宏包后，右侧的非终结符将链接到对应的定义处（如果其定义存在）。

当使用字符标记时，可使用 `\h<(non-terminal)>` 的形式显示使用。在定义的左侧使用时，被解释为设置该非终结符的超链接位置；在定义的右侧使用时，被解释为链接到这个非终结符的定义处。可以显式使用 `\BNFAnchor` 或 `\BNFref` 来表示上述的两种类型。

```
hyper = (true|false)
hyper-color = {<(颜色)>}
```

初始值: **false**

 texbnf/hyper
 texbnf/hyper-color

`hyper` 控制是否使用默认使用超链接而无需显示使用 `\h`。

`hyper-color` 控制超链接的颜色。未给定时，使用超链接默认的颜色。

`\BNFNP77` 超链接的使用与否也受 `hyper` 选项控制。

例 68

```
\begin{latexbnf}[hyper, hyper-color=purple]
<glue> ::= <optional signs><internal glue>
| <dimen><stretch><shrink>
<stretch> ::= "plus"<dimen> | "plus"<fil dimen> | <optional spaces>
<shrink> ::= "minus"<dimen> | "minus"<fil dimen> | <optional spaces>
<fil dimen> ::= <optional signs><factor><fil unit><optional spaces>
<fil unit> ::= "fil" | <fil unit>"l"
<muglue> ::= <optional signs><internal muglue>
| <mudimen><mustretch><mushrink>
<mustretch> ::= "plus"<mudimen> | "plus"<fil dimen> | <optional spaces>
<mushrink> ::= "minus"<mudimen> | "minus"<fil dimen> | <optional spaces>
\end{latexbnf}
```

```
(glue) → (optional signs)(internal glue)
| (dimen)(stretch)(shrink)
(stretch) → plus(dimen) | plus(fil dimen) | (optional spaces)
(shrink) → minus(dimen) | minus(fil dimen) | (optional spaces)
(fil dimen) → (optional signs)(factor)(fil unit)(optional spaces)
(fil unit) → fil | (fil unit)l
(muglue) → (optional signs)(internal muglue)
| (mudimen)(mustretch)(mushrink)
(mustretch) → plus(mudimen) | plus(fil dimen) | (optional spaces)
(mushrink) → minus(mudimen) | minus(fil dimen) | (optional spaces)
```

完全等价的一个写法是：

例 69

```
\begin{latexbnf}[hyper, hyper-color=purple]
\BNFItem \BNFN{glue}\is\BNFN{optional signs}\BNFN{internal glue}
\alt\BNFN{dimen}\BNFN{stretch}\BNFN{shrink}
\BNFItem \BNFN{stretch}\is\BNFT{plus}\BNFN{dimen}\alt\BNFT{plus}\BNFN{fil
↪ dimen}\alt\BNFN{optional spaces}
\BNFItem \BNFN{shrink}\is\BNFT{minus}\BNFN{dimen}\alt\BNFT{minus}\BNFN{fil
↪ dimen}\alt\BNFN{optional spaces}
\BNFItem \BNFN{fil dimen}\is\BNFN{optional signs}\BNFN{factor}\BNFN{fil
↪ unit}\BNFN{optional spaces}
\BNFItem \BNFN{fil unit}\is\BNFT{fil}\alt\BNFN{fil unit}\BNFT{l}
\BNFItem \BNFN{muglue}\is\BNFN{optional signs}\BNFN{internal muglue}
\alt\BNFN{mudimen}\BNFN{mustretch}\BNFN{mushrink}
\BNFItem
↪ \BNFN{mustretch}\is\BNFT{plus}\BNFN{mudimen}\alt\BNFT{plus}\BNFN{fil
↪ dimen}\alt\BNFN{optional spaces}
\BNFItem
↪ \BNFN{mushrink}\is\BNFT{minus}\BNFN{mudimen}\alt\BNFT{minus}\BNFN{fil
↪ dimen}\alt\BNFN{optional spaces}
\end{latexbnf}
```

```
texbnf/format
texbnf/Nformat
texbnf/Iformat
texbnf/Oformat
texbnf/Tformat
texbnf/clear-all-format
```

```
format = {(code)}
Tformat = {(code)}
clear-all-format
```

初始值：\ttfamily
不可设置值

设置格式。

$\text{Nleft} = \{\langle code \rangle\}$	初始值: $\backslash\text{ensuremath}\{\backslash\text{langle}\}, \rangle$	$\text{texbnf}/\text{Nleft}$
$\text{Nright} = \{\langle code \rangle\}$	初始值: $\backslash\text{ensuremath}\{\backslash\text{rangle}\}, \rangle$	$\text{texbnf}/\text{Nright}$
$\text{I} = \{\langle code \rangle\}$	初始值: $\backslash\text{ensuremath}\{\backslash\text{longrightarrow}\}, \rightarrow$	$\text{texbnf}/\text{Tleft}$
$\text{O} = \{\langle code \rangle\}$	初始值: $\backslash\text{cus@mathrule}, $	$\text{texbnf}/\text{Tright}$
设置 $\backslash\text{BNFN}_{\text{P77}}$ 、 $\backslash\text{BNFT}_{\text{P77}}$ 左右的符号, $\backslash\text{BNFI}_{\text{P77}}$ 、 $\backslash\text{BNFO}_{\text{P77}}$ 的替换符号。		texbnf/N
		texbnf/T
		texbnf/O
		texbnf/I
$\text{label-prefix} = \{\langle \text{前缀} \rangle\}$		$\text{texbnf}/\text{label-prefix}$
$\text{label-suffix} = \{\langle \text{后缀} \rangle\}$		$\text{texbnf}/\text{label-suffix}$

当设置超链接锚点时，会写入 \label，使用 label-prefix 和 label-suffix 可在 \label 名中添加〈前缀〉和〈后缀〉。使用 \BNFref 时，也需正确添加它们。

§ 6 ref 库

ref 库提供交叉引用的一些额外功能。

本库改进了 \IfPageOdd_{P9} 和 \IfAbsPageOdd_{P9}, 使得它们在任何位置都有效。

<pre>\IfLabelOdd {<label>} {<true>} {<false>}</pre>	<pre>\IfLabelOdd</pre>
<p>判断这个 $\langle label \rangle$ 是否定义在奇数页。当 $\langle label \rangle$ 不存在时使用 $\langle false \rangle$。</p> <p>$\langle label \rangle$ 所在页码必须以阿拉伯数字显示，否则使用 $\langle false \rangle$ 分支。</p>	
<pre>\HyperRef {<label>} {<material>} \HyperRef * {<label>} {<material>} \HyperLink {<target>} {<material>} \HyperLink * {<target>} {<material>}</pre>	<pre>\HyperRef \HyperLink</pre>

`hyperref` 宏包提供了 `\hyperref` 和 `\hyperlink` 命令，用于链接到 `label` 或 `hyper target`，但是这两个命令的参数不能包含特殊的文本，如不能包含 `verbatim` 文本和 `\parbox` 等。`\HyperRef`^{[E_{p.79}](#)} 和 `\HyperLink`^{[E_{p.79}](#)} 用于解决这个问题。

`\HyperRef`[Ⓔ]`{\langle label \rangle}{\langle material \rangle}` 和 `\hyperref[\langle label \rangle]{\langle text \rangle}` 效果完全一样。其作用是把 `\langle text \rangle` 链接到 `\langle label \rangle` 所在的那个地方，但可以包含特殊文本，仅能在特殊的位置断行。

`\HyperLink[target]{text}` 和 `\hyperlink[target]{text}` 完全一样，其作用是把 *text* 链接到 *target* 所在的那个地方，但可以包含特殊文本，仅能在特殊的位置断行。

它们还有带星号的用法。功能相同，但 `<material>` 中可以有特殊的文本，也可以正常换行。

需注意比如虽然 `(material)` 可以包含特殊文本，如 `parbox` 等，但其本身并不能作为其它命令的参数。

另见 \cus_ref_label_box:nn_{p4s}、\cus_ref_target_box:nn_{p4s}和 \cus_ref_label_shbox:nn_{p5s}、\cus_ref_target_shbox:nn_{p5s}。

除了上面所说的简单的用法，这两个命令还支持用可选参数设置 (*material*) 的宽度、高度及对齐方式。

\HyperRef
\HyperLink

```
\HyperRef    {<label>} [<width>] <material>
\HyperRef *  {<label>} [<width>] <material>
\HyperRef    {<label>} [<width>] [<vpos>] [<height>] [<inner pos>] <material>
\HyperRef *  {<label>} [<width>] [<vpos>] [<height>] [<inner pos>] <material>
```

当不带星号使用时，相当于把 $\langle material \rangle$ 放在 `minipage` 中， $\langle material \rangle$ 的宽度设置为 $\langle width \rangle$ ，但可以包含特殊文本。当带星号使用时，相当于把 $\langle material \rangle$ 放在 `varwidth` 中， $\langle material \rangle$ 的最大宽度为 $\langle width \rangle$ ，也可以包含特殊文本。

它们的可选参数正如 `\parbox` 的可选参数那样：

- $\langle vpos \rangle$ 表示垂直位置，可选值为 `b`、`c`、`t`，分别表示对齐底部基线、居中对齐、对齐顶部基线，默认为居中对齐；
- $\langle height \rangle$ 表示盒子的高度，如果不设置，则为盒子的自然高度，
- $\langle inner pos \rangle$ 表示如果设置的 $\langle height \rangle$ 过大时，盒子的内容在盒子内的垂直位置，可选值为 `b`、`c`、`t`、`s`，分别表示置于盒子底部、居中、置于顶部、垂直分散对齐，默认为垂直分散对齐。

另见 `\collectn_minipage:Nnnnnnw`^{P.87}、`\collectn_varwidth:Nnnnnnw`^{P.87}。

虽然这两个命令不能配置 $\langle material \rangle$ ，但 `box` 库提供了 `\fparbox`^{P.83} 和 `\fvarbox`^{P.83} 命令可以设置盒子外框，见第 5.7.3 小节及例 70。

§ 7 box 库

`box` 库提供额外的一些盒子。

5.7.1 paracol 环境

`paracol` 提供了另一种多栏盒子。它可以控制文字出现在某栏，也可以手动对齐各栏的文字，常用于排版多语言对照文本。

\startparacol
\stopparacol

```
\startparacol [<paracol keyval>]
...
\stopparacol
```

使用 `paracol` 宏包排版多栏文字。

\switchcolumn

```
\switchcolumn
\switchcolumn *
\switchcolumn [<col>]
\switchcolumn [<col>] *
\switchcolumn [<col>] * [<heading>]
```

切换至第 $\langle col \rangle$ 栏，栏以 0 开始计数。若不给出 $\langle col \rangle$ 则切换至下一栏。

如果 `*` 给出，则先构建先前的文字，将它们对齐，然后接下来的文字的对齐位置为这些已经对齐了的文字的底部。

如果 $\langle heading \rangle$ 给出，则作为通栏文字插入。

\thecolumn

当前栏数，以 0 开始计数。

paracol/cols
paracol/numleft
paracol/paired

```
cols      = {<总栏数>}
numleft   = {<在左页的栏数>}
paired    = true|false
```

初始值：`true`

$\langle cols \rangle$ 设置多栏的总栏数。`paracol` 环境支持多栏分布在左右两页，通过 $\langle numleft \rangle$ 设置在左侧页的栏数。在此情况下， $\langle paired \rangle$ 用于设置这左右两页的页码是否相同。

```
heading = {\text}
```

```
paracol/heading
```

在多栏文本前插入通栏文字 $\langle text \rangle$ 。

```
column-ratio      = {\langle r_0, r_1, \dots, r_k \rangle} [\langle r_0', r_1', \dots, r_k' \rangle]
column-ratio-left = {\langle r_0, r_1, \dots, r_k \rangle}
column-ratio-right = {\langle r_0', r_1', \dots, r_k' \rangle}
```

```
paracol/column-ratio
paracol/column-ratio-left
paracol/column-ratio-right
```

设置每栏宽度的占比。 $\langle column-ratio \rangle$ 的可选参数和 $\langle column-ratio-right \rangle$ 设置的是右侧页的栏。

每栏的实际宽度为 $r_i \times (\text{\textwidth} - (n - 1) \text{\columnsep})$ 。未给出的栏的宽度为剩余的宽度除以剩余的栏数。

其作用类似于 `\columnratio`。

每个环境开始时，它总被重置。

```
column-width      = {\langle s_0, s_1, \dots, s_k \rangle} [\langle s_0', s_1', \dots, s_k' \rangle]
column-width-left = {\langle s_0, s_1, \dots, s_k \rangle}
column-width-right = {\langle s_0', s_1', \dots, s_k' \rangle}
```

```
paracol/column-width
paracol/column-width-left
paracol/column-width-right
```

设置每栏的宽度和间距。 $\langle column-width \rangle$ 的可选参数和 $\langle column-width-right \rangle$ 设置的是右侧页的栏。

s_i 的形式为 \hat{w}_i 或 \hat{w}_i/\hat{g}_i ，这里 \hat{w}_i 、 \hat{g}_i 为一个 glue 或 dim，或为空表示 $\hat{w}_i = \text{\fill}$ ， $\hat{g}_i = \text{\columnsep}$ 。未给出的假定为空。

如果给出的总宽度大于 `\textwidth`，则每个宽度按比例缩放，使得总宽度为 `\textwidth`。

其作用类似于 `\setcolumnwidth`。

每个环境开始时，它总被重置。

例如，假设 `\textwidth=360pt`，`\columnsep= S=20pt`，则

s_0, s_1, s_2	w_0	g_0	w_1	g_1	w_2 (in pt)
50pt/20pt, 100pt/40pt, 150pt	50	20	100	40	150
50pt, 100pt/2\columnsep, 150pt	50	S	100	2S	150
50pt/\fill, 100pt/2\fill, 150pt	50	$(1/3) \cdot 60$	100	$(2/3) \cdot 60$	150
, 2\fill/2\columnsep, 3\fill	$(1/6) \cdot 300$	S	$(2/6) \cdot 300$	2S	$(3/6) \cdot 300$
50pt/20pt, 50pt plus 1fil/40pt, 50pt plus 2fil	50	20	$50 + (1/3) \cdot 150$	40	$50 + (2/3) \cdot 150$
5pt/2pt, 10pt/4pt, 15pt	$10 \cdot 5$	$10 \cdot 2$	$10 \cdot 10$	$10 \cdot 4$	$10 \cdot 15$
100pt/40pt, 200pt/80pt, 300pt	$0.5 \cdot 100$	$0.5 \cdot 40$	$0.5 \cdot 200$	$0.5 \cdot 80$	$0.5 \cdot 300$

保存了当前栏宽度的 dim 寄存器。

```
\columnwidth
```

```
twosided = \page|p|column|c|margin|m|background|b|all|none>
```

初始值: [all](#)

```
paracol/twosided
```

使用 `twoside` 排版特性。在奇偶页输出不同的效果。

page|p 正如 `twoside` 文档类选项控制的那样。

column|c 在偶数页逆序输出各栏。

margin|m 切换边注的位置。

background|b 切换背景。

all 设置上述为真。

none 设置上述为假。注意若要单独设置某个选项，需要先使用 `none` 将它们都设置为假。

```
paracol/marginpar-threshold
paracol/marginpar-threshold-left
paracol/marginpar-threshold-right
```

```
marginpar-threshold      = {\<k>} [\<k'\>]
marginpar-threshold-left = {\<k>}
marginpar-threshold-right = {\<k'\>}
```

设置前 k 栏的边注放在左侧。

注意，边注的位置还受到 `twosided` 选项中的 `margin` 的控制和 `\reverse-marginpar` 的控制。

```
paracol/counter-global
paracol/counter-local
```

```
counter-global = {\<counter list>} | *
counter-local  = {\<counter list>}
```

默认情况下，除了 `page` 计数器外，其它计数器的值的改变仅作用于某一栏。可以使用 `counter-global` 设置对各栏可见。`counter-local` 的效果则相反。

设置 `counter-global` 为 `*` 时，相当于设置所有计数器。

它们的设置是全局的。相当于 `\globalcounter`、`\localcounter` 命令。

```
\definethecounter
```

```
\definethecounter {\<counter>} {\<col>} {\<rep>}
```

将第 $\langle col \rangle$ 栏的 `\the\<counter>` 修改为 $\langle rep \rangle$ 。

```
\synccounter
\syncallcounter
```

```
\synccounter {\<counter>}
\syncallcounter
```

将计数器 $\langle counter \rangle$ （或所有计数器）在此栏的值同步到其它栏。

```
paracol/column-sep-rule
```

```
column-sep-rule = {\<长度>}
```

设置栏间分割线的宽度。

```
paracol/before
paracol/before+
paracol/after
paracol/after+
```

```
before = {\<code>}
before+ = {\<code>}
```

设置在环境开始或结束时要执行的代码。

```
paracol/preamble
```

```
preamble      = {\<code>}
preamble [\<col>] = {\<code>}
```

设置第 $\langle col \rangle$ 栏开始时执行的代码。 $\langle col \rangle$ 为 -1 表示通栏文字执行前执行的代码。如果 $\langle col \rangle$ 未给出则为 -1 。

```
\columncolor
\normalcolumncolor
\colseprulecolor
\normalcolseprulecolor
```

```
\columncolor [\<mode>] {\<color>} [\<col>]
\normalcolumncolor [\<col>]
\colseprulecolor [\<mode>] {\<color>} [\<col>]
\normalcolseprulecolor [\<col>]
```

设置每栏文字的颜色即栏间竖线的颜色。`\normalcolumncolorP82`、`\normalcolseprulecolorP82` 用于恢复为原始颜色。

如果在环境外使用，则 $\langle col \rangle$ 未给出时设置的是第 0 栏的颜色；在环境内使用，则设置的是当前栏的颜色。

设置是全局的。

```
paracol/contents
```

```
contents = {\<file>} {\<col>}
```

将第 $\langle col \rangle$ 栏的章节添加到目录文件 $\langle file \rangle$ 。 $\langle file \rangle$ 只能是 `toc`、`lof`、`lot` 之一。

设置是全局的。

关于 `paracol` 宏包的其它用法见 `paracol` 宏包文档。

5.7.2 multicolumns/framed=lfbox

本库提供为多栏文字的外框提供了 `lfbox` 可选值。表示用 `longfbox` 宏包的 `\lfbox` 作为盒子外框。`multicolumns/framed-options`_{P24} 可以使用 `longfbox` 宏包提供的选项。

5.7.3 \fparbox_{P83} 和 \fvarbox_{P83}, 可设置外框的命令

`longfbox` 宏包提供了使用 CSS 属性名来设置盒子外框的命令 `\lfbox` 和环境 `longfbox`, 但 `\lfbox` 只能包含水平模式的内容, 这里提供了 `\fparboxP83` 和 `\fvarboxP83` 以补全这一缺点。

```
\fparbox <{width}> <{material}>
\fparbox [<vpos>] [<height>] [<inner pos>] <{width}> [<longfbox options>] <{material}>
\fvarbox <{width}> <{material}>
\fvarbox [<vpos>] [<height>] [<inner pos>] <{width}> [<longfbox options>] <{material}>
```

```
\fparbox
\fvarbox
```

`\fparboxP83` 把 `<material>` 封装进 `minipage` 中, `\fvarboxP83` 把 `<material>` 封装进 `varwidth` 中。

`<vpos>`、`<height>`、`<inner pos>`、`<width>`、`<material>` 选项的含义前面已提过多次, 如在 `\HyperRefP80` 命令的说明中。`<longfbox options>` 为 `longfbox` 中可用的键值选项。

```
\hypersetup{linkcolor=red} 链接到
\HyperRef{sec:fparbox-fvarbox}{\fparbox[t]{3cm}
[border-color={}] {可以分段的\par \verb|\lfbox|}}
\HyperRef{sec:fparbox-fvarbox}{\fvarbox[c]{3cm}
[border-color={}] {可以分段的\par \verb|\lfbox|}}
```

例 70

链接到

可以分段的
`\lfbox`

可以分段的
`\lfbox`

§ 8 math 库

```
\delsize <{real}> <left>
\delsize <{real}> <middle>
\delsize <{real}> <right>
```

```
\delsize
\delsize
\delsize
\delsize
\delsize
\delsize
\delsize
```

正如 `\bigl`、`\bigm` 和 `\bigr` 那样, 但如上几个命令可以设置括号的大小。

`\delsize..` 设置的 `<real>` 为数学模式下左括号 (高度的倍数。`\bigsize..` 设置的 `<real>` 为数学模式下左括号 (高度的倍数的 1.2 倍。

例如, `\bigl` 相当于 `\bigsize{1}`。

§ 9 counter 库

本库定义了与计数器相关的一些命令。

```
\ensuretwodigits <{计数器}>
```

```
\ensuretwodigits ☆
\ensurethreedigits ☆
\ensurefourdigits ☆
```

类似于 `\arabic`, 只不过确保它至少输出 2 (或 3、4) 个数, 不足的补 0。

`\MakePerPage`

```

\MakePerPage          {\counter clist}
\MakePerPage [initial value] {\counter clist}
\MakePerPage *         {\int /count list}
\MakePerPage * [initial value] {\int /count list}

```

不带星号的命令的作用为, 在新的一页重设 *<counter clist>* 中的计数器为 *<initial value>*, 这个初值默认为 0。

带星号的命令的作用为, 在新的一页重设 *<int /count list>* 中的寄存器为 *<initial value>*, 这个初值默认为 0。

<counter clist> 中的计数器不是必须用 `\newcounter` 来定义。

`\IfIsCounterTF`

```

\IfIsCounterTF {<tl>} {\true} {\false}

```

判断 *<tl>* 是不是计数器, 必须是使用 `\newcounter` 定义的。

T_EXhackers note: 一个计数器并不一定直接由 `\newcounter` 定义, 也可能由其它命令间接调用 `\newcounter`。

只要 *<tl>* 在 `\cl@ckpt` 中, 则结果为 *<true>*, 否则为 *<false>*。

`\RecordTotalCounters`
`\IfRecordTotalCounterTF`

```

\RecordTotalCounters {\counter clist}
\IfRecordTotalCounterTF {\counter} {\true} {\false}

```

记录计数器的总次数。可使用 `\thetotal<counter>s` 来获取这个值。当加载了 **xs-space** 宏包后, `\thetotal<counter>x` 会在 `\thetotal<counter>s` 后加上 `\xspace`。

`\thetotal<counter>s` 一般用于正文中。只能在加载了 `.aux` 文件后才能使用。即可以在 `begindocument` (或 `\AtBeginDocument`)、`begindocument/end` 钩子中使用。

§ 10 pdf 库

本库提供一些与 PDF 文件的特性相关的功能。它们只能在使用了 `\DocumentMetadata` 之后使用。

默认情况下, 在 T_EX 中直接导入 PDF 文件会丢失该 PDF 的超链接信息, 使用本库可以解决此问题。相较于直接使用 **newpax** 宏包, 若导入的 PDF 文件中有很多超链接, 本库提供的 `\includegraphicspax`^[P84] 速度会快很多。

`pdf/pax`
`pdf/pax+`

```
pax = {\pdf file list}
```

为了能够保留导入的 PDF 文件的超链接信息, 需要为这些 PDF 文件预先生成一些必要的辅助文件。本选项用于设置需要保留超链接信息的那些 PDF 文件。

如果使用 Lua_TE_X 则可以自动生成。其它引擎需要开启 `shell-escape` 功能。仅在导言区设置才有效。

`pdf/graphics-path`

```
graphics-path = {\path clist}
```

在查找需要保留超链接的 PDF 文件时, 设置查找路径。如果没有设置, 则使用 `\setgraphicspath`^[P4] 设置的。

`\includegraphicspax`

```

\includegraphicspax * [{graphics options}] {\filename}

```

用此命令导入 PDF 文件可以保留超链接。需编译两次。

```
redefine = <true|false>
```

初始值: **false**

pdf/redefine

是否将 `\includegraphics` 重定义为 `\includegraphicspaxFS4`。

第六章 可单独加载的宏包

§ 1 collectn

```
\collectn_verbatim:Nnw <tl> {<code>} <token> <tokens> <token>
\collectn_verbatim:Nnw <tl> {<code>} {<balanced tokens>}
```

```
\collectn_verb:Nnw
\l_collectn_long_verbatim_bool
```

向后以 `verbatim` 的形式收集内容, 将其保存至 `<tl>` 中, 再执行 `<code>`。`<token>` 或 `<balanced tokens>` 不能在命令的参数里。

`\l_collectn_long_verbatim_boolFS4` 控制是否接受长的 `verbatim` 文本 (包含 `\par` 的)。

在 `<tokens>` 或 `<balanced tokens>` 中, 字母的类别码不能为 1、2、10。

结果 `<tl>` 包含的字符其类别码有三种, 为 10、12、13 (在 (u)pTeX 引擎下, 仅 Unicode 编码小于等于 255 的字符有此特性), 且 `^^M` 的类别码为 13。

```
\collectn_varwidth:nnw {<vertical pos>} {<maximum width>}
<contents> \collectn_varwidth_end:
```

```
\collectn_varwidth:nnw
\collectn_varwidth_end:
```

`<contents>` 是可变宽的, 最大宽度为 `<maximum width>`; 基线的位置为 `<vertical pos>`, 可选值为 `b`、`c`、`t`, 分别表示对齐底部基线、居中对齐、对齐顶部基线。

该命令是对 `varwidth` 环境的封装。

```
\collectn_set_varwidth:Nnnw <box> {<vertical pos>} {<maximum width>}
<contents> \collectn_set_varwidth_end:
```

```
\collectn_set_varwidth:Nnnw
\collectn_set_varwidth_end:
\collectn_gset_varwidth:Nnnw
\collectn_gset_varwidth_end:
```

设置 `<box>` 为一个包含 `<contents>` 的最大宽度为 `<maximum width>` 的水平盒子。

```
\collectn_set_vbox_width:Nnw <box> {<width>}
<content> \collectn_set_vbox_width_end:
```

```
\collectn_set_vbox_width:Nnw
\collectn_set_vbox_width_end:
\collectn_set_vbox_varwidth:Nnw
\collectn_set_vbox_varwidth_end:
\collectn_gset_vbox_width:Nnw
\collectn_gset_vbox_width_end:
\collectn_gset_vbox_varwidth:Nnw
\collectn_gset_vbox_varwidth_end:
```

`\collectn_set_vbox_width:NnwFS5` 与 `\vbox_set:Nw` 类似, 把 `<content>` 保存到 `vbox <box>` 中, 该 `<box>` 的宽度为 `<width>`。它和 `minipage` 环境相似。

`\collectn_set_vbox_varwidth:NnwFS5` 则是设置 `<content>` 的最大宽度为 `<width>`。它和 `varwidth` 环境相似。

这 `<box>` 可以使用 `\vsplit` 或 `\vbox_set_split_to_ht:Nnn` 来分割。

```
\collectn_box:NNnw <box> <primitive box cs> {<code>} <material>
```

```
\collectn_box:Nnw
```

先将 `<material>` 保存到 `<box>` 中, 此盒子为 `<primitive box cs>`, 可为 `\hbox`、`\vbox`、`\vtop` 之一。之后再使用 `<code>` 处理。

`<material>` 可以是 `{<horizontal/vertical mode material>}`, 也可以有 `<box specification>`。左右括号可以是隐式的。

这种方式与 `\peek_charcode_remove:NTF` 类似。另见 `collectbox` 宏包。

`<code>` 与 `\collectn_box:NnwFS5` 在同一个组中执行。

注意: `\hbox:n`、`\vbox:n`、`\vbox_top:n` 并不分别等于 `\hbox`、`\vbox`、`\vtop`。

例 71

```

\ExplSyntaxOn
\cs_set:Npn \myfbox
{
  \collectn_box:NNnw \l_tmpa_box \tex_hbox:D
  { \fbox { \box_use_drop:N \l_tmpa_box } }
}
\ExplSyntaxOff
\myfbox{\verb|\myfbox| 与 \verb|\fbox|}
\myfbox{\parbox{3cm}{可分段的\par fbox}}

```

\myfbox 与 \fbox	可分段的 fbox
-----------------	--------------

另见例 85。

\collectn_value:Nnw\collectn_value:Nnw <register> {<code>} <value>

将 <value> 保存到 <register> 中，再使用 <code> 处理。<value> 必须确实可以保存到 <register> 中。

<code> 与此命令在同一个组中执行。

例 72

```

\ExplSyntaxOn
\cs_set:Npn \showintval
{ \collectn_value:Nnw \l_tmpa_int { 值为 $ \int_use:N \l_tmpa_int $ } }
\ExplSyntaxOff
\showintval 3
\showintval -310
\showintval "3FA
\showintval \shellescape

```

值为 3 值为 -310 值为 1018 值为 2

\collectn_width:Nnnw
\collectn_minipage:Nnnw
\collectn_varwidth:Nnnw
\collectn_width:Nnnw <box> {<code>} {<width>} <material>

先使用类似于 minipage (varwidth) 的处理方式处理 <material>，然后将它保存到 vbox <box> 中，再使用 <code> 处理。这 <box> 的宽度（或最大宽度）为 <width>。

<material> 可以是 {<vertical mode material>}，正如 \vbox {<vertical mode material>} 那样。左右括号可以是隐式的。

<code> 与这些命令在同一个组中执行。

`\collectn_minipage:Nnnnw`^{P.86} 是 `\collectn_width:Nnnnw`^{P.86} 的另一个名字。

保存的垂直盒子可以用 \vsplit 或 \vbox_set_split_to_ht:NNn 来分割。

\collectn_width:Nnnnw
\collectn_minipage:Nnnnw
\collectn_varwidth:Nnnnw
\collectn_width:Nnnnw <box> {<code>} {<vpos>} {<width>} <material>

先使用类似于 minipage (varwidth) 的处理方式处理 <material>，然后将它保存到 hbox <box> 中，再使用 <code> 处理。这 <box> 的宽度为 <width>，基线的位置为 <vpos>，可选值为 b、c、t，分别表示底部基线、居中、顶部基线，默认为底部基线。

<material> 可以是 {<vertical mode material>}，正如 \vbox {<vertical mode material>} 那样。左右括号可以是隐式的。

<code> 与这些命令在同一个组中执行。

`\collectn_minipage:Nnnnw`^{P.86} 是 `\collectn_width:Nnnnw`^{P.86} 的另一个名字。

例 73

```

\ExplSyntaxOn
\cs_set:Npn \myparbox #1#2
{
  \collectn_width:Nnnnw \l_tmpa_box
  { \fbox { \box_use_drop:N \l_tmpa_box } } {#1} {#2}
}
\cs_set:Npn \myvarbox #1#2
{
  \collectn_varwidth:Nnnnw \l_tmpa_box
  { \fbox { \box_use_drop:N \l_tmpa_box } } {#1} {#2}
}
\ExplSyntaxOff
\myparbox{t}{5cm}{一个可以包含 \verb|\verb| 的\par
定宽 \verb|\fbox|}
\myvarbox{b}{5cm}{一个可以包含 \verb|\verb| 的\par
变宽 \verb|\fbox|}

```

一个可以包含 \verb 的
定宽 \fbox

一个可以包含 \verb 的
变宽 \fbox

```

\collectn_width:Nnnnnnw <box> {<code>}
{<vpos>} {<height>} {<inner pos>} {<width>} <material>

```

```

\collectn_width:Nnnnnnw
\collectn_minipage:Nnnnnnw
\collectn_varwidth:Nnnnnnw

```

前述命令的完整形式。

- *<vpos>* 表示垂直位置，可选值为 b、c、t，分别表示对齐底部基线、居中对齐、对齐顶部基线，默认为对齐底部基线；
- *<height>* 表示盒子的高度，如果不设置，则为盒子的自然高度，
- *<inner pos>* 表示如果设置的 *<height>* 过大时，盒子的内容在盒子内的垂直位置，可选值为 b、c、t、s，分别表示置于盒子底部、居中、置于顶部、垂直分散对齐，默认为垂直分散对齐。

<material> 可以是 {<vertical mode material>}，正如 \vbox {<vertical mode material>} 那样。左右括号可以是隐式的。

<code> 与这些命令在同一个组中执行。

```

\collectn_hbox_auto:Nnnw <box> {<code>} {<spec>} {<content>}

```

```

\collectn_hbox_auto:Nnnw
\collectn_width_auto:Nnnw
\collectn_minipage_auto:Nnnw
\collectn_varwidth_auto:Nnnw

```

根据 *<spec>* 自动使用合适的命令。

`\collectn_hbox_auto:Nnnw` 的 *<spec>* 为 \makebox 的前两个参数：[<width>] [<pos>]。

其它三个命令的 *<spec>* 为 \parbox 的前四个参数：[<vpos>] [<height>] [<inner pos>] [<width>]。仅当只有 *<width>* 给出时，所保存的盒子为垂直盒子，且可以使用 \vsplit 或 \vbox_set_split_to_ht:NNn 来分割该盒子。

```

\collectn_if_in:nTF {<token list>} {<true code>} {<false code>}

```

```

\collectn_if_in:nTF

```

判断下一个记号是否在给定的 *<token list>* 中。

无法判断显式或隐式的左、右括号和空格。

```

\collectn_delimited_inline:NNn <token1> <token2> {<inline code>}
\collectn_delimited_variable:NNNn <token1> <token2> <tl> {<code>}

```

```

\collectn_delimited_inline:NNn
\collectn_delimited_variable:NNNn

```

获取接下来的以 *<token₁>* 和 *<token₂>* 定界的参数。若不存在，则为 \q_no_value。*<inline code>* 可以用 #1 获取这个参数，*<tl>* 用于保存这个参数。

`\collectn_delimited_any_inline:nn`

`\collectn_delimited_any_inline:nn {⟨paired tokens⟩} {⟨inline code⟩}`

判断接下来的定界的参数的定界符是否在 $\langle paired\ tokens \rangle$ 之中。 $\langle inline\ code \rangle$ 可使用 2 个参数，第一个为此定界符在 $\langle paired\ tokens \rangle$ 中的位置，第二个为它的值。若不存在，则第一个值为 0，第二个值为 `\q_no_value`。

`\collectn_delimited_all_inline:nn`

`\collectn_delimited_all_inline:nn {⟨paired tokens⟩} {⟨inline code⟩}`

对于 $\langle paired\ tokens \rangle$ 给出的定界符对，逐一考察是否接下来的参数以此定界符对定界。

`\l_collectn_delimited_strip_bool`

用于控制上面四个命令在获取定界的参数时，若其参数整个用 $\{\}$ 包裹，是否移除此 $\{\}$ 。

```
\ExplSyntaxOn \ttfamily
\bool_set_false:N \l_collectn_delimited_strip_bool
\collectn_delimited_inline:NNn [ ] { \tl_to_str:n { |#1| } } [{bracket}] ;
\collectn_delimited_inline:NNn [ ] { \tl_to_str:n { |#1| } } \scan_stop: ;
\par
\collectn_delimited_any_inline:nn { [ ] ( ) } { \tl_to_str:n { |#1,#2| } }
  [{bracket}] ;
\collectn_delimited_any_inline:nn { [ ] ( ) } { \tl_to_str:n { |#1,#2| } }
  ({brace}) ;
\par
\collectn_delimited_all_inline:nn { [ ] ( ) } { \tl_to_str:n { |#1| } }
  (brace) [{bracket}] ;
\collectn_delimited_all_inline:nn { [ ] ( ) } { \tl_to_str:n { |#1| } }
  ({brace}) ;
\ExplSyntaxOff
```

```
|{bracket}|;|\q_no_value |;
|1,[bracket]|;|2,{brace}|;
|[{bracket}]{brace}|;|{\q_no_value }{brace}}|;
```

例 74

`\collectn_lohi:Nnw`
`\collectn_lohis:Nnw`

`\collectn_lohi:Nnw ⟨tl⟩ {⟨code⟩} ⟨content⟩`

判断接下来的一些字符是否是数学上下标。`\collectn_lohi:Nnw例 75` 获取一组数学上下标（即数学下标和上标每个最多获取一个）；`\collectn_lohis:Nnw例 75` 可获取任意多个上下标。把这些上下标存储到 $\langle tl \rangle$ 里，然后执行 $\langle code \rangle$ 。

$\langle tl \rangle$ 由形如 $\{ \sim \langle text \rangle \}$ 或 $\{ _ \langle text \rangle \}$ 的项组成。其中 \sim 和 $_$ 的类别码分别为 7 和 8。

它们会自动展开后面的内容，并且自动移除空格和 `\relax`。

它们只会匹配类别码为 7 和 8 的显式或隐式字符。若加载了 `underscore` 宏包，则 $_$ 的类别码会修改为 13，在文本模式下无法匹配，但在数学模式下可以。

```
\ExplSyntaxOn
\collectn_lohi:Nnw \l_tmpa_tl { \tl_to_str:N \l_tmpa_tl }
  \c_math_subscript_token \scan_stop: { a } ~ { b }
\par
\collectn_lohis:Nnw \l_tmpa_tl { \tl_to_str:N \l_tmpa_tl }
  \scan_stop: \c_math_superscript_token \scan_stop: { a }
  \scan_stop: \sb { b } \sp { c } ~ { d }
\ExplSyntaxOff
```

例 75


```
{_ {a}} {^ {b}}
{^ {a}} {_ {b}} {^ {c}} {^ {d}}
```

```
\collectn_scan_keyword:nTF {<keyword>} {<true>} {<false>}
```

```
\collectn_scan_keyword:nTF
\collectn_scan_keywordscs:nTF
```

判断接下来的一些字符是否匹配 $\langle keyword \rangle$ ，若匹配则移除它们。为了检测后面的文字是否匹配 $\langle keyword \rangle$ ，它会自动展开后面的命令。

$\langle keyword \rangle$ 被转为类别码为 12 的记号列（包括空格），两端的空格不会被移除。`\collectn_scan_keyword:n` 忽略大小写，`\collectn_scan_keywordscs:n` 不忽略大小写。

```
\collectn_set_keyword:Nn <str> {<keyword>}
```

```
\collectn_set_keyword:Nn
\collectn_set_keyword:(No|NV|Nv|Ne|cn|co|cV|cv|ce)
```

把 $\langle keyword \rangle$ 处理成可供使用的记号列，然后将其保存至 $\langle str \rangle$ 。

先对 $\langle keyword \rangle$ 执行 `\tl_to_str:n`，然后把所有空格替换为类别码为 12 的空格。

```
\collectn_scan_keyword:NTF <keyword str> {<true>} {<false>}
```

```
\collectn_scan_keyword:NTF
\collectn_scan_keyword:cTF
\collectn_scan_keywordscs:NTF
\collectn_scan_keywordscs:cTF
```

判断接下来的一些字符是否匹配 $\langle keyword \rangle$ ，若匹配则移除它们。为了检测后面的文字是否匹配 $\langle keyword \rangle$ ，它会自动展开后面的命令。

$\langle keyword str \rangle$ 需用 `\collectn_set_keyword:NnP89` 设置。

```
\collectn_scan_keywords:nTF {<keyword clist>} {<true>} {<false>}
```

```
\collectn_scan_keywords:nTF
\collectn_scan_keywordscs:nTF
```

判断接下来的一些字符是否匹配 $\langle keywords clist \rangle$ 中的某一项，若匹配则移除它们。忽略大小写。为了检测后面的文字是否匹配 $\langle keywords clist \rangle$ 中的项，它会自动展开后面的命令。

对 $\langle keyword list \rangle$ 中的每一项，先移除两端的空格和空项，然后转为类别码为 12 的记号列（包括空格）。

它不会移除 $\langle keywords clist \rangle$ 中重复的项，若有重复的项，则以最后出现的那个为准。

在 $\langle true \rangle$ 和 $\langle false \rangle$ 中可使用 `\l_collectn_keywords_intP90` 来获取匹配的 $\langle keyword \rangle$ 的位置，`\l_collectn_keywords_strP90` 获取匹配的 $\langle keyword \rangle$ 。

```
\collectn_set_keywords:NTF <tl> {<keyword clist>}
```

```
\collectn_set_keywords:Nn
\collectn_set_keywords:(No|NV|Nv|Ne|cn|co|cV|cv|ce)
```

对 $\langle keyword clist \rangle$ 中的每一项先移除两端的空格，再应用 `\collectn_set_keyword:NnP89`，然后将这些项保存至 $\langle tl \rangle$ 。

它不会移除 $\langle keywords clist \rangle$ 中重复的项，若有重复的项，则以最后出现的那个为准。

```
\collectn_set_keywords_from_seq:NN <tl> <seq>
```

```
\collectn_set_keywords_from_seq:NN
\collectn_set_keywords_from_seq:(Nc|cn|cc)
```

对 $\langle seq \rangle$ 中的每一项应用 `\collectn_set_keyword:NnP89`，然后将这些项保存至 $\langle tl \rangle$ 。

它不会移除重复的项，若有重复的项，则以最后出现的那个为准。

```
\collectn_scan_keywords:N\TF
\collectn_scan_keywords:c\TF
\collectn_scan_keywordscs:N\TF
\collectn_scan_keywordscs:c\TF
```

```
\collectn_scan_keywords:N\TF <keywords tl> {\true} {\false}
```

判断接下来的一些字符是否匹配 $\langle keywords\ tl \rangle$ 中的某一项，若匹配则移除它们。为了检测后面的文字是否匹配 $\langle keywords\ tl \rangle$ 中的项，它会自动展开后面的命令。

$\langle keywords\ tl \rangle$ 需用 $\backslash\collectn_set_keywords:Nn_{\text{P.89}}$ 或 $\backslash\collectn_set_keywords_from_seq:Nn_{\text{P.89}}$ 设置。

```
\l_collectn_keywords_int
\l_collectn_keywords_str
```

$\langle keywords\ clist/tl \rangle$ 中匹配的 $\langle keyword \rangle$ 的位置和值。若不匹配则位置为 0，值为空。

§ 2 lt3ekeys

本宏包扩展了 l3keys 的部分功能。

6.2.1 定义键

6.2.2 设置键

6.2.3 lt3ekeys-elkernel

本宏包为 L^AT_EX 和 L^AT_EX3 的一些有用的内部命令提供公共接口。

```
\elkernel_arg_to_keyvalue:Nnn
```

```
\elkernel_arg_to_keyvalue:Nnn <result> {\key} {\arg}
```

检查 $\langle arg \rangle$ 是否为键值对，若不是，则把 $\langle key \rangle$ 作为键， $\langle arg \rangle$ 作为值。将结果保存到 $\langle result \rangle$ 中。

检查是否为键值对的方法和 ltcmd 的 = spec 一致。

```
\elkernel_if_date_at_least:nn\TF
```

```
\elkernel_if_date_at_least:nn\TF {\date_1} {\date_2} {\true} {\false}
```

测试 $\langle date_1 \rangle$ 是否等于或晚于 $\langle date_2 \rangle$ 。年月日可用 \ 或 - 分隔，但不可混用。

```
\ExplSyntaxOn
\tl_to_str:N \fmtversion;
\elkernel_if_date_at_least:nn\TF { \fmtversion } { 2024/06/01 } { t } { f }
\ExplSyntaxOff
```

2023-11-01;f

例 76

6.2.4 定义命令——lt3ekeyscmd

本宏包实现了一个与 \DeclareDocumentCommand 类似的命令，提供了更多常用的功能。

以下称使用 \DeclareDocumentCommand 定义的命令为 *document-cmd*，称使用 $\backslash\keysdeclarecmd_{\text{P.91}}$ 定义的命令为 *ekeys-cmd*。

```
\ekeysdeclarecmd <cmd> {\arg spec} {\code}
\ekeysdeclarecmd * <cmd> {\arg spec} {\code}
\ekeys_declare_cmd:Nnn <cmd> {\arg spec} {\code}
\ekeys_declare_cmd_x:Nnn <cmd> {\arg spec} {\code}
```

```
\DeclareEKeysCommand
\ekeysdeclarecmd
\ekeys_declare_cmd:Nnn
\ekeys_declare_cmd_x:Nnn
\l_ekeys_cmd_defaults_bool
```

类似于 `\DeclareDocumentCommand`, 部分 `<arg spec>` 未实现, 增加了一些 `<arg spec>`。

`\ekeysdeclarecmdP91` 等于 `\ekeys_declare_cmd:NnnP91`。`\ekeys_declare_cmd_x:NnnP91` 完全展开 `<code>`。

`\DeclareEKeysCommandP91` 是 `\ekeysdeclarecmdP91` 的另一个名字。

`<arg spec>` 可以是数字, 此时相当于 `\cs_generate_from_arg_count:NNnn <cmd> \cs_set_protected:Npn {\arg spec} {\code}`。

`<arg spec>` 不为数字 (或空) 时定义的命令称为 `ekeys-cmd`。

最大的参数数量为 9。

此命令定义的命令均为使用长参数。即它们定义的命令总是 `\protected\long`。

目前, 在向后寻找可选参数时, 会忽略空格, 但未来可能会修改为不忽略空格, 因此不宜在可选参数前加上空格 (但必需参数 `m`、`r`、`R` 总忽略空格)。如 `\foo_{a}_{o}` 应写为 `\foo {a}{o}`, 假定 `arg spec` 为 `mo`。

在定义 `ekeys-cmd` 时, 若设置 `\l_ekeys_cmd_defaults_boolP91` 为真, 则可选参数的默认值可以引用其它实参 (称为 “执行可选参数默认值引用替换”), 否则 `<arg spec>` 中的参数不能引用其它实参, 如 `r[] D(){#1}` 中的 `#1` 只是普通文本。`\ekeysdeclarecmdP91` 不带星号的版本设置 `\l_ekeys_cmd_defaults_boolP91` 为假, `\ekeysdeclarecmdP91` 带星号的版本设置 `\l_ekeys_cmd_defaults_boolP91` 为真。`\l_ekeys_cmd_defaults_boolP91` 只有在使用上述三个命令时才会生效。注意: 若默认参数之间互相循环引用, 则在执行时会出错, 但 `document-cmd` 则不会出错。如命令的 `spec` 为 `D(){#2} D<>{#1}`, 则命令为 `ekeys-cmd` 会出错, 而为 `document-cmd` 不会出错。

`ekeys-cmd` 可以用于 `\ShowCommand`、`\NewCommandCopy` 中, 也支持 `cmd` 钩子, 限制条件和 `document-cmd` 一致。

```
\DeclareEKeysCollector <collector> {\arg spec}
\DeclareEKeysCollector * <collector> {\arg spec} <tl> {\do code}
```

```
\DeclareEKeysCollector
\ekeysdeclarecollector
```

定义和 `ekeys-cmd` 类似的命令, `<arg spec>` 的参数数量可以超过 9 个。这种命令称之为 `ekeys-collector`。

```
\ekeyscollectorarg {\arg number}
```

```
\ekeyscollectorarg ☆
```

完全展开为第 `<arg number>` 个参数。如果没有该值, 则为 `\q_no_value`, 可以使用 `\IfQuarkNoValueTFP94` 检测。非正整数都无效。它只能用于 `ekeys-collector` 的 `<do code>` 中。

注意: 不能使用 `f` 展开, 因为值可能为 `\q_no_value`, 它会造成无限递归。

```
\DeclareEKeysCollector \faa { m m @w{ ( ) [] } m m m m m m m }
\faa\tmp{\meaning\tmp.} 12 [[a]](b(b)) 56789ABCD.

\faa\tmp{\edef\tmp{\ekeyscollectorarg{10}}\meaning\tmp.}
12 [[a]](b(b)) 56789ABCD.
```

例 77

```
\DeclareEKeysCollector*\fbb { m m @w{ ( ) [] } m m m m m m } \tmp
↪ {\meaning\tmp.}
\fbb 12 [[a]](b(b)) 56789ABCD.
```

```
macro:->{1}{2}{b(b)}{[a]}{5}{6}{7}{8}{9}{A}{B}.CD.
macro:->A.CD.
macro:->{1}{2}{b(b)}{[a]}{5}{6}{7}{8}{9}{A}{B}.CD.
```

`\ekeysgenerategrabber`

```
\ekeysgenerategrabber <grabber tl> {\arg spec}
```

从 `<arg spec>` 生成 `grabber`。生成的 `grabber` 可以用于 `\ekeyscollectargsP92` 命令中。

预先生成 `grabber` 可以避免重复生成以节省时间。

默认最大可用的参数个数为 `\e@alloc@top`，它远远超过 9。在 pdf_LA_TE_X 和 X_LA_TE_X 中为 $2^{15} - 1 = 32767$ ，在 Lua_LA_TE_X 和 (u)p_LA_TE_X 中为 $2^{16} - 1 = 65535$ 。可以通过 `\l_ekeys_cmd_max_args_intP92` 修改。

`\ekeys_grabber_args_do:Nnn`

```
\ekeys_grabber_args_do:Nnn <tl> <grabber tl> {\do code}
```

首先收集 `<grabber tl>` 所需的参数，每个参数都用 `{ }` 括起来，将其保存到 `<tl>` 中，然后执行 `<do code>`。

此命令不会执行可选参数默认值引用替换。

`\ekeys_collect_args_do:Nnn`
`\ekeyscollectargs`

```
\ekeys_collect_args_do:Nnn <tl> <ekeys-cmd OR grabber tl> {\do code}
```

首先收集 `<ekeys-cmd>` 所需的参数或 `<grabber tl>` 所需的参数，每个参数都用 `{ }` 括起来，将其保存到 `<tl>` 中，然后执行 `<do code>`。

只有当第二个参数为 `ekeys-cmd` 时，才会此执行可选参数默认值引用替换。

`\ekeys_collect_args_do:Nnn`
`\ekeyscollectargs*`

```
\ekeys_collect_args_do:Nnn <tl> {\arg spec} {\do code}
```

首先根据 `<arg spec>` 收集所需的参数，每个参数都用 `{ }` 括起来，将其保存到 `<tl>` 中，然后执行 `<do code>`。

最大可用的参数个数为 `\e@alloc@top`，它远远超过 9。在 pdf_LA_TE_X 和 X_LA_TE_X 中为 $2^{15} - 1 = 32767$ ，在 Lua_LA_TE_X 和 (u)p_LA_TE_X 中为 $2^{16} - 1 = 65535$ 。可以通过 `\l_ekeys_cmd_max_args_intP92` 修改。

此命令相当于先用 `\ekeysgenerategrabberP92` 生成 `grabber`，然后再使用 `\ekeyscollectargsP92`。

此命令不会执行可选参数默认值引用替换。

`l_ekeys_cmd_max_args_int`

最大的可收集的参数数目。

`\ekeys_cmd_undefine:N`

```
\ekeys_cmd_undefine:N <ekeys-cmd>
```

将 `<ekeys-cmd>`（局部地）设置为未定义。如果不是 `ekeys-cmd` 则保持不变。对 `ekeys-cmd` 和 `ekeys-collector` 都有效。

`\ekeysmakeglobal`
`\ekeys_cmd_global:N`

```
\ekeysmakeglobal <ekeys-cmd>
```

把局部定义的 `<ekeys-cmd>` 改为全局定义的。`\ekeysdeclarecmdP91` 总是局部地定义命令。如果不是 `ekeys-cmd` 则保持不变。

```
\__ekeys_if_cmd:NTF <cs> {\<true code>} {\<false code>}
```

```
\__ekeys_if_cmd:NTF *
```

简单判断一个控制序列是否为 *ekeys-cmd*。

可用的参数类型为：

m 标准的必须的参数；

r `r<token1><token2>`；

R `R<token1><token2>{\<default>}`；

o 可选参数；

O `O{\<default>}`；

d `d<token1><token2>`；

D `D<token1><token2>{\<default>}`；

s 可选的星号，如果给出则为 `\BooleanTrue`，否则为 `\BooleanFalse`；

t `t<token>`，如果 `<token>` 给出，则为 `\BooleanTrue`，否则为 `\BooleanFalse`；

k `k<keyword>`，如果 `<keyword>` 给出，则为 `\BooleanTrue`，否则为 `\BooleanFalse`；

定义命令时，忽略 `<keyword>` 的类别码，执行时，仅比较字符，忽略类别码（活动字符除外，它将导致失配）和大小写的区别；另见 `\collectn_scan_keyword:nTFES9`；

t `t{\<token pairs>}`，它占用两个参数，用于确定多个定界符对是否有一个给出了，第一个为定界符对所在的位置（如果未出现则为 0），第二个为它的值；

T `T{\<token pairs>}{\<default>}`，它占用两个参数，用于确定多个定界符对是否有一个给出了，第一个为定界符对所在的位置（如果未出现则为 0），第二个为它的值（如果未出现则为 `<default>`）；

p `p{\<tokens>}`，如果出现在 `<tokens>` 中，则给出所在的位置，否则给出 0；

P `P{\<tokens>}{\<defaults>}`，`<defaults>` 的长度必须为 `<tokens>` 的长度加一，用出现在 `<tokens>` 中的位置索引 `<defaults>`（`<defaults>` 的索引从 0 开始），`p{*-+}` 相当于 `P{*-+}{0123}`；

e `e{\<tokens>}`，使用 `w` 参数实现，每个符号可以重复使用；

E `E{\<tokens>}{\<defaults>}`，使用 `w` 参数实现，每个符号可以重复使用；

w `w{\<token pairs>}`，智能匹配多个 `d` 类型的参数，从而可以以任意顺序给出参数的值；

W `W{\<token pairs>}{\<defaults>}`，智能匹配多个 `D` 类型的参数，从而可以以任意顺序给出参数的值；

g 可选的以 `{` 开始的参数，`{` 可以是显式或隐式的；若后面跟着的不是 `{` 而是 `\relax`，则移除这个 `\relax`；

G `G{\<default>}`，可选的以 `{` 开始的参数，`{` 可以是显式或隐式的；若后面跟着的不是 `{` 而是 `\relax`，则移除这个 `\relax`；

l `{` 前的内容，如没有 `{` 则出错；

u `u{\<tokens>}`，`<tokens>` 前的内容，`<tokens>` 会被移除，如没有 `<tokens>` 则出错；

U `U{\<tokens1>}{\<tokens2>}`，`<tokens1>` 前的内容，移除 `<tokens1>` 并留下 `<tokens2>`，如没有 `<tokens1>` 则出错；

v 以 `verbatim` 的形式读取文字，该参数所含字符的类别码的可能值为 10、12、13（在 (u)p_{TE}X 引擎下，仅 Unicode 编码小于等于 255 的字符有此特性）；且 `^^M` 的类别码为 13；

未实现的参数类型为 `b`, `+`, `!`, `>`, `=`, 使用它们仅给出警告, 不会报错。

除非另有说明否则上述参数类型的参数中均不能出现显式或隐式的空格 (`\`), 宏变量字符 (`#`)、左右括号 (`{ }`), 即 `d<#`、`t\bgroup` 等均为错误用法。

`u` 和 `U` 的参数可以有空格, 如可以 `u{ }`, 必须有括号。

带有左右定界符的参数在使用时不能嵌套, 如 $\langle arg\ spec \rangle$ 为 `d<>`, 则 `\foo<a>` 的参数为 `a<b`, 要得到 `a`, 必须写 `\foo<\{a\}>`。

`lt3ekeysext` 宏包对其进行了进一步扩展, 例如支持定义可嵌套的定界符以及更高的执行效率。见第 6.2.5 小节。

`\ekeys_cmd_name: *`

完全展开为 `ekeys-cmd` 的名称。仅能用在 `ekeys-cmd` 执行的 $\langle code \rangle$ 中。

`\IfQuarkNoValueTF *`
`\IfQuarkNoValueT *`
`\IfQuarkNoValueF *`

 $\backslash\text{IfQuarkNoValueTF } \langle arg \rangle \{ \langle true\ code \rangle \} \{ \langle false\ code \rangle \}$

判断 $\langle arg \rangle$ 是否为 `\q_no_value`。

`\NumberCase *`
`\@numbercase *`

 $\backslash\text{NumberCase } \langle integer \rangle \{ \{ \langle case_0 \rangle \} \{ \langle case_1 \rangle \} \dots \{ \langle case_n \rangle \} \} \{ \langle else \rangle \}$
 $\backslash\text{@numbercase } \langle integer \rangle ; \{ \langle case_0 \rangle \} \{ \langle case_1 \rangle \} \dots \{ \langle case_n \rangle \} ; \{ \langle else \rangle \}$

根据 $\langle integer \rangle$ 的值选择对应的 $\langle case_i \rangle$, 如果 $integer > n$ 或 $integer < 0$ 则使用 $\langle else \rangle$ 。

`\BooleanFalse` 相当于 0, `\BooleanTrue` 相当于 1。

展开两次即可得到结果。

例 78

```
\ExplSyntaxOn
\keys_declare_cmd:Nnn \foo { p{!@*~+} } { 位置为: #1. }
\keys_declare_cmd:Nnn \fee { p{!@*~+} }
{ \NumberCase {#1} { {错误!} } { 给出 \tl_item:nn {!@*~+} {#1}. } }
\keys_declare_cmd:Nnn \fii { P{!@*~+}{ {错误}{!}{@}{*}{-}{+} } } {
  ↳ 给出: #1. }
\ExplSyntaxOff

\foo @ \par
\foo ? \par
\fee @ \par
\fee ? \par
\fii @ \par
\fii ?
```

位置为: 2.
 位置为: 0?
 给出 @.
 错误! ?
 给出: @.
 给出: 错误.?

例 79

```
\keysdeclarecmd \foo { t{ ( ) [] <> } } {位置为: #1, 值为: #2.}
\foo <c> \foo (p) \foo\relax
```

位置为: 3, 值为: c。位置为: 1, 值为: p。位置为: 0, 值为: -NoValue-。

`w` 和 `W` 中 (包括 `e` 和 `E`) 的定界符对可以重复使用, 但在第一个相同的情况下第二个也必须相同。如支持 `w{ [] () [] }`、`w{ () [] < > }`, 但不支持 `w{ [] [] }`。


```
\ekeysdeclarecmd \foo { w{ [] () <> } } {\{#1;#2;#3\}}
\ekeysdeclarecmd \fee { w{ [] [] () } } {\{#1;#2;#3\}}
```

例 80

```
\foo <a> (b) [c] , \foo (a) [b] <c> ,
\fee [a] (b) [c] , \fee [a] [b] (c) , \fee (a) [b] [c] ,
```

```
{c;b;a} , {b;a;c} , {a;c;b} , {a;b;c} , {b;c;a} ,
```

t、T、w、W 的每对的第二个还能为空或空格。例如 `w{ _{} ^{} }` 相当于 `e{ _{} }`，例如支持 `w{ [] () _{} ^{} :{} }`。但第一个不能为空或空格，如不支持 `w{ {} _{} }`。

t 和 T 用于仅需要一个参数的情况，这些定界符对只会检测一次，而 w 和 W 则支持多个，可以以任意顺序给出。

```
\ekeys_exp_not_braced:n {\text}
```

展开为 `\exp_not:n{\text}`。

```
\ekeys_exp_not_braced:n *
\ekeys_exp_not_braced:(o|v|f|e) *
```

```
\ekeys_cmd_after_declare:n {\code}
```

定义完本命令后执行 `<code>`。在定义命令的外面使用是无效的。

```
\ekeys_cmd_after_declare:n
\ekeys_cmd_after_declare:e
```

6.2.5 定义命令扩展——lt3ekeysext

lt3ekeysext 进一步扩展了 lt3ekeyscmd 的功能。

为 `\ekeysdeclarecmd` 增加了几个前缀：

- # 将其后的那个参数类型标记为需要特殊处理（如果可以的话），使用此前缀能（大大）提高 *ekeys-cmd* 的执行速度，但有轻微的副作用（一般很难发生，详见后文）；支持 T、p、P、u、U、e、E、w、W；
- @ 将其后的那个参数类型标记为需要嵌套（如果可以的话）；
- & 将其后的那个参数类型标记为保留等价的原始输入；支持 r、R、o、O、d、D、s、t、p、k、t、T、g、G、e、E、w、E；

在使用 # 前缀时，参数在其定界符为控制序列的时候，定义和使用 *ekeys-cmd* 时 `\escapechar` 的值必须为相同的正数。如，当定义 `\foo` 时 `\escapechar` 为 `\`，则使用 `\foo` 时也必须为 `\`。受影响的参数类型为 t、T、p、P、e、E、w、W。

使用 t、T、e、E、w、W 时，设置 & 将自动设置 #。

对于 s、t 和 k 参数，使用 & 前缀，在检测到存在此 *<token>*（或 *<keyword>*）时，会把这个参数设置为该 *<token>*（或 *<keyword>*），否则这个参数为空。使用其它几个支持 & 前缀的参数时，若检测到有此定界符，则该参数会包含此定界符，否则该参数为 *<default>*。

```
\newcommand{\abdel}[5]{#1#3#4#2#5}
\ekeysdeclarecmd\ab{ p{\big\Big\bigg\Bigg*} &t{ \{\} [] () || } }
{\IfNoValueTF{#3}{NAN}% 如果没有所列的括号，输出 NAN
  {\NumberCase{#1} {
    {\abdel\left\right}
    {\abdel\bigl\bigr} {\abdel\Bigl\Bigr}
    {\abdel\biggl\biggr} {\abdel\Biggl\Biggr} {\abdel{}{}}
  }}
  #3}
}
```

例 81


```
$ \ab[\dfrac{12}{}]{ } $, $ \ab\Big(\dfrac{12}{}){ } $, $ \ab*|\dfrac{12}{}| $, $ \ab? $
```

$$\left[\frac{1}{2}\right], \left(\frac{1}{2}\right), |\frac{1}{2}|, NAN?$$

```
\ekeysdeclarecmd\foo{ &s &t{ ( ) [] } &w{ !{ } +{ } } &k{p_t} }
{ 1.\{#1\}, 2.\{#2|#3\}, 4.\{#4\}\{#5\} 6.\{#6\} }
```

例 82

```
\ttfamily
```

```
{\catcode\_ =13 \foo *[b]+plus !{mu} p_t} , % 此时 _ 为活动字符
```

```
{\catcode\_ =8 \foo *[b]+plus !{mu} p_t} , % 此时 _ 为下标
```

```
\foo (b)!mu ,
```

```
1.{*}, 2.{2|[b]}, 4.{!mu}{+plus } 6.{ } p_t , 1.{*}, 2.{2|[b]},
4.{!mu}{+plus } 6.{p_t} , 1.{ }, 2.{1|(b)}, 4.{!m}{-NoValue-} 6.{ }
u ,
```

执行效率从慢到快大致如下：无 # 且为 t、T、w、W 的 *ekeys-cmd* < 无 # 且有 @ 的 *ekeys-cmd* < *document command* < 带 # 和 @ 的 *ekeys-cmd* < 带 # 的 *ekeys-cmd* < `\ekeys_grabber_args_do:NNn` < `__ekeys_grabber_args_do:NNn`。

为 `\ekeysdeclarecmd` 增加了几个参数类型：

c *c{<collect spec>}*。特殊的保存方式；必须的参数；

C *C{<allocated collect spec>}*。特殊的保存方式；必须的参数；

K *K{<scanner-and-args>}*。自定义的参数扫描器。

c 和 C 用 TeX 寄存器来保存各种值，设置这些值的方式和 TeX 相似。前者会自动分配一个新的寄存器，而 C 使用已经分配好的寄存器，这是它们的唯一区别。这两个参数类型的实参为所分配的寄存器，一般情况下不应直接修改这个寄存器。

如

```
\ekeysdeclarecmd\foo{ c{i} m }{[\the#1,\detokenize{#2}]}
\ttfamily
\foo -12 {a}; \quad \foo -12 \relax; \quad \foo -12\relax;
\foo \interval{12+8*(-2)}\relax; \foo \numexpr 12+8*(-2)\relax;
```

例 83

```
[-12,a]; [-12,\relax ]; [-12,\relax ]; [-4,\relax ]; [-4,;]
```

定义了一个命令 `\foo`，第一个参数存储一个整数，第二个参数为通常的参数。注意到给出第一个参数时与通常的使用方式很不相同，这是 TeX 中为寄存器赋值的语法，每个类型都遵循各自的语法规则，详细用法见 *The TeXbook*。

<collect spec> 可用的值如下：

i 为 int (count) 寄存器赋值；

d 为 dim (dimen) 寄存器赋值；

s 为 skip 寄存器赋值；

m 为 muskip 寄存器赋值；

t 为 toks 寄存器赋值；

b *<extra spec>*，为一个水平盒子赋值；*<extra spec>* 可以为 “*” 表示可自由设置盒子宽度；也可为 *<width>* [*<pos>*]，正如 `\makebox` 的前两个参数；

w *<extra spec>*, 为一个固定宽度的盒子赋值; 类似于把参数放在一个 `minipage` 里; *<extra spec>* 可以为 `[<vpos>][<height>][<inner pos>]{<width>}`, 正如 `\parbox` 的前四个参数一样。若只给出 *<width>* 则为垂直盒子, 且可以对其使用 `\vsplit`, 否则为水平盒子;

v *<extra spec>*, 为一个有最大宽度的盒子赋值; 类似于把参数放在一个 `varwidth` 里; *<extra spec>* 可以为 `[<vpos>][<height>][<inner pos>]{<width>}`, 正如 `\parbox` 的前四个参数一样。若只给出 *<width>* 则为垂直盒子, 且可以对其使用 `\vsplit`, 否则为水平盒子。

注意 *<extra spec>* 的外侧不能有花括号, 除非是只有 *<width>* 这个必须参数。并且可选参数之间不能有额外的空格。如 `b{[*]}`、`b[3cm]`、`w{3cm}`、`w[c]{3cm}` 可以, `b{[3cm]}`、`b{[3cm][1]}`、`w{[c]{3cm}}` 不可以。

前 5 个类型, 即 `i`、`d`、`s`、`m`、`t`, 的实参可以作为 `LTEX3` 函数的 `V` 变体的参数。

<allocated collect spec> 是一个已经分配的寄存器加上 *<collect spec>*, 若 *<collect spec>* 的类型没有必须参数, 则可以省略不写, 只写寄存器, 但寄存器必须和类型一致。

<collect spec> 和 *<allocated collect spec>* 中不能引用其它实参, 因为它们并非默认值。

```
\newdimen\tmpadim
\keysdeclarecmd\foo { C{\tmpadim} c{i} c{b[2cm][c]} }
{ [\the#1, \the#2, \fbox{\box#3}] }
\foo 12pt -1 {a a}
\foo \dimexpr 5cm+12pt\relax \numexpr 3+4*(1-2)\relax {a a}
```

例 84

[12.0pt, -1, a a] [154.26378pt, -1, a a]

```
\keysdeclarecmd \myfbox { c{b} } {\fbox{\box#1}}
\myfbox{\verb|\myfbox| 与 \verb|\fbox|}
\myfbox{\parbox{3cm}{可分段的\par fbox}}
```

例 85

\myfbox 与 \fbox 可分段的
fbox

```
\keys_cmd_new_scanner:nnnpn {\scanner} {\argument numbers}
{\initial action} \parameter list {\scanner action}
\keys_cmd_new_scanner:nnpn {\scanner} {\argument numbers}
\parameter list {\scanner action}
```

```
\keys_cmd_new_scanner:nnnpn
\keys_cmd_new_scanner:nnpn
\keys_cmd_new_scanner:nnnpn
\keys_cmd_new_scanner:nnpx
```

为参数类型 `K` (局部地) 定义新的扫描器 *<scanner>*。该扫描器为 `ekeys-cmd` 增加 *<argument numbers>* 个参数。

<initial action> 为定义 `ekeys-cmd` 时要执行的操作。可以使用参数, 分别为 `ekeys-cmd` 的名称, 给该扫描器的额外的参数, 此 `ekeys-cmd` 已有的参数个数, 是否为 `raw` 参数 (使用了 `&`), 以及是否为 `nested` 参数 (使用了 `@`)。

注意, 直接在 *<initial action>* 中定义 `ekeys-cmd` 和 `ekeys-collector` 是不行的。但可以在其中使用 `\keys_cmd_after_declare:npgs`, 先定义完本命令后再定义新的命令。

给扫描器的额外的参数指的是: 移除 *<scanner-and-args>* 的首尾空格后, 若它以一对 `{ }` 开始, 则这个花括号里的内容就是 `scanner`, 之后的内容移除掉两端的空格后就是额外的参数; 否则, 就是 `scanner` 就是第一个花括号之前的内容 (移除掉两端的空格), 额外的参数就是第一个花括号及其之后的记号。如若 *<scanner-and-args>* 为 `\{a_b\}[v]_j`, 则 `scanner` 为 `a_b`, 参数为 `[v]_j`; 若 *<scanner-and-args>* 为 `\fo_o{da}_[b]`, 则 `scanner` 为 `fo_o`, 参数为 `{da}_[b]`。

<scanner action> 为 `ekeys-cmd` 执行到该扫描器时需要执行的操作。 *<parameter*

list) 为需要的形参列表。⟨*scanner action*⟩ 中, 使用 `\ekeys_cmd_add_args:nP100` 为 *ekeys-cmd* 添加实参。⟨*scanner action*⟩ 完成时, 必须执行 `\ekeys_cmd_scanner_end: P99`。

注意 `\ekeys_cmd_name: P94` 不能在 ⟨*initial action*⟩ 中使用 (可以用 #1 替代), 但可以在 ⟨*scanner action*⟩ 中使用。另见 `\ekeys_cmd_args: P100`。

`\ekeys_cmd_name: P94` 与 ⟨*initial action*⟩ 的第一个可用的参数相同, `\ekeys_cmd_args: P100` 与 ⟨*scanner action*⟩ 的第三个可用参数相同。

```
\ekeysnewscanneralias
\ekeys_cmd_new_scanner_alias:nn
\ekeys_cmd_new_scanner_alias:nnn
```

```
\ekeysnewscanneralias {⟨alias⟩} {⟨scanner definition⟩}
\ekeysnewscanneralias {⟨alias⟩} [⟨spec name⟩] {⟨part spec⟩}
\ekeys_new_scanner_alias:nn {⟨alias⟩} {⟨scanner definition⟩}
\ekeys_new_scanner_alias:nnn {⟨alias⟩} {⟨spec name⟩} {⟨part spec⟩}
```

(局部地) 定义扫描器的别名。

⟨*scanner definition*⟩ 是用于在 K 的 ⟨*scanner-and-args*⟩ 里的内容。⟨*spec name*⟩ 是标识参数类型的字母, ⟨*part spec*⟩ 是此类型具体的一个。见例 87。

当使用 `K{u⟨args⟩}` 时, 只使用 ⟨*args*⟩ 的第一项, 多余的部分会放到 ⟨*scanner definition*⟩ 或 ⟨*part spec*⟩ 后面。

有几个预定义的扫描器:

`norelax`, 移除一个 `\relax`, 在向后寻找这个 `\relax` 时忽略空格;

`norelax!`, 移除一个 `\relax`, 在向后寻找这个 `\relax` 时不忽略空格;

`?⟨preprocessor spec⟩`, 参数预处理器。⟨*preprocessor spec*⟩ 用法为:

- `{⟨ekeys-cmd arg spec⟩}`, 使用 ⟨*ekeys-cmd arg spec*⟩ 获取后面的参数, 转化为带花括号的标准参数;
- `{⟨ekeys-cmd arg spec⟩}{⟨scanner code⟩}`, 使用 ⟨*ekeys-cmd arg spec*⟩ 获取后面的参数, 并用 ⟨*scanner code*⟩ 替换这些参数。⟨*scanner code*⟩ 必须包含 `\ekeys_cmd_scanner_end: P99`, 且在它后面的内容会被 *ekeys-cmd* 重新读取;
- `{⟨ekeys-cmd arg spec⟩}{⟨scanner code⟩}{⟨text⟩}`, 同上, ⟨*text*⟩ 会放在要读取的参数之前。

`u u{⟨alias⟩}`, 使用由 `\ekeysnewscanneralias P98` 设置的别名。

define 可以用它来在定义 *ekeys-cmd* 时, 进一步的自定义参数获取方式。用法为 `K{define⟨define spec⟩}`。⟨*define spec*⟩ 可以为:

- `{⟨definition⟩}`, 它不占用任何参数。主要用作向后检查, 或展开, 也可以向输入中添加额外的内容。需要在 ⟨*definition*⟩ 的合适位置添加 `\ekeys_cmd_add_args:nP100` 和 `\ekeys_cmd_scanner_end: P99`;
- `{⟨numbers⟩}{⟨parameters⟩}`, 它占用 ⟨*numbers*⟩ 个参数。根据 ⟨*parameters*⟩ 向后获取参数。⟨*parameters*⟩ 的参数数量不能少于 ⟨*numbers*⟩。这是 `\def` 的 *ekeys-cmd* 接口;
- `{⟨numbers⟩}{⟨parameters⟩}{⟨definition⟩}`, 同上。注意: 当 *numbers* ≥ 0 时, 会在 ⟨*definition*⟩ 后面自动加上 `\ekeys_cmd_add_args:nP100` 和 `\ekeys_cmd_scanner_end: P99`;
- `{*⟨numbers⟩}{⟨parameters⟩}{⟨definition⟩}`, ⟨*numbers*⟩ 与 ⟨*parameters*⟩ 没有关系。但需要在 ⟨*definition*⟩ 的合适位置添加 `\ekeys_cmd_add_args:nP100` 和 `\ekeys_cmd_scanner_end: P99`, 且添加的参数必须和 ⟨*numbers*⟩ 一致。

lohi 它获取一组数学上下标, 占用 2 个参数, 第一个为下标, 第二个为上标。如果不存在, 则为使用默认值。设置默认值是通过给扫描器的额外的参数来设置: `K{lohi⟨defaults⟩}`, 其中 ⟨*defaults*⟩ 为:

- `{⟨defaultlo⟩}{⟨defaulthi⟩}`, 设置下标和上标的默认值;
- `{⟨default⟩}`, 设置上下标的默认值;

- 空，表示上下标使用特殊的标记：-NoValue-。

它支持 `raw` 修饰符，即可以自动添加 `_` 和 `^`，但默认值不会自动添加这两个符号。

```
\ekeysdeclarecmd\faa{ K{lohi} }{[#1.#2]}
\ekeysdeclarecmd\fbb{ K{lohi{}} }{[#1.#2]}
\ekeysdeclarecmd\fcc{ &K{ lohi {_x}{^y} } } {[$\int #1#2$]}

\faa ^b_a; \faa \relax _a ^\relax b; \faa ^b; \faa ?; \par
\fbb ^b; \fbb _a^b; \fbb ?; \par
\fcc _a^b; \fcc _a; \fcc ?;
```

.....

[a.b]; [a.b]; [-NoValue-.b]; [-NoValue.-NoValue-]?;
 [.b]; [a.b]; [.]?;
 \int_a^b ; \int_a^y ; \int_x^y ?;

例 86

```
\ekeysnewscanneralias{math-style}[p]
{p{\displaystyle\textstyle\scriptstyle\scriptscriptstyle}}
\ekeysnewscanneralias{scan-bra-ket}{define{2}{<#1|#2>}}

\DeclareEKeysCommand \foo { &K{u{math-style}} K{u{scan-bra-ket}} }
{{#1\left<#2\middle|#3\right>}}
$ \foo<a|b> $ \quad $ \foo\displaystyle<\sum|\prod> $.
```

.....

$\langle a|b \rangle$ $\langle \sum | \prod \rangle$.

例 87

除了 `?` 参数扫描器外，还有一个预处理指示符 `?`，用法为 `?{\preprocessor action}`。预处理器指示符可以看成是 `?` 参数扫描器的预先构建的版本。

实际上，`scanner` 的参数数量不仅可以通过 `<argument numbers>` 设置，还可以在 `<initial action>` 中直接修改 `\l_ekeys_cmd_scanner_args_int`^{§99} 来设置参数数量。

参数扫描器完成时，必须执行该命令。放在此命令之后的内容会添加到输入中，可供后面的参数解析。

如果不是用在扫描器或预处理器内，`\ekeys_cmd_scanner_end`^{§99} 会出错，而 `\EKeysEndPreprocessor`^{§99} 则什么也不做。预处理器和自定义的参数扫描器必须包含它们之一。

`\l_ekeys_cmd_scanner_args_int`

`\ekeys_cmd_scanner_end:`
`\EKeysEndPreprocessor`

```
\ExplSyntaxOn
\ekeys_cmd_new_scanner_alias:nn { replace-keyword-to-num }
{
  define
  {
    \collectn_scan_keywords:nTF { true, false, on, off, yes, no }
    {
      \int_if_odd:nTF \l_collectn_keywords_int
      { \ekeys_cmd_scanner_end: 1~ }
      { \ekeys_cmd_scanner_end: 0~ }
    }
    { \ekeys_cmd_scanner_end: -1~ }
  }
}
\DeclareEKeysCommand \foo { K{u{replace-keyword-to-num}} u{~} } {[#1]}
\ExplSyntaxOff
\foo TRUE; \foo on; \foo false; \foo ;
```

.....

例 88

[1]; [1]; [0]; [-1];

`\ekeys_cmd_add_arg:n`
`\ekeys_cmd_add_arg:(o|V|v|f|e)`

`\ekeys_cmd_add_arg:n {<argument>}`

为 `ekeys-cmd` 添加 1 个实参。⟨*scanner action*⟩ 中添加的实参总数量必须和定义扫描器时设置的数目一致。

`\ekeys_cmd_add_args:n`
`\ekeys_cmd_add_args:(o|V|v|f|e)`

`\ekeys_cmd_add_args:n { {<argument1>} ... {<argumentn>} }`

为 `ekeys-cmd` 添加 n 个实参。⟨*scanner action*⟩ 中添加的实参总数量必须和定义扫描器时设置的数目一致。

`\ekeys_cmd_register_cs:N`
`\ekeys_cmd_register_cs:c`

`\ekeys_cmd_register_cs:N <auxiliary function>`

仅当 ⟨*auxiliary function*⟩ 不能在不同命令之间共享时才需注册，否则可以直接在外层定义，不必置于 ⟨*initial action*⟩ 之内。

注册命令的主要作用是使用 `\ekeys_cmd_undefine:NP92` 时，把这些注册的命令也设置为未定义。

`\ekeys_cmd_scanner_undefine:n`
`\ekeys_cmd_scanner_alias_undefine:n`

`\ekeys_cmd_scanner_undefine:n {<scanner>}`
`\ekeys_cmd_scanner_alias_undefine:n {<scanner alias>}`

将 ⟨*scanner*⟩ 或 ⟨*scanner alias*⟩ 局部地设置为未定义。

`\ekeys_cmd_args: *`

完全展开为到目前为止此 `ekeys-cmd` 已经使用了的参数数目。仅能用于 ⟨*scanner action*⟩。可与 `\ekeys_cmd_name:P94` 配合使用。

请看下面两个例子了解它们的用法。

例 89

```
\ExplSyntaxOn
\ekeys_cmd_new_scanner:nnnpn { my/if-pt } { 1 }
{
  \ekeys_cmd_register_cs:c { #1-#3-my/if-pt/is-row }
  \bool_new:c { #1-#3-my/if-pt/is-row }
  \bool_set:cn { #1-#3-my/if-pt/is-row } {#4}
}
{
  \bool_set_eq:Nc \l_tmpa_bool
  { \ekeys_cmd_name: - \ekeys_cmd_args: - my/if-pt/is-row }
  \collectn_scan_keyword:nTF { pt }
  {
    \bool_if:NTF \l_tmpa_bool
    { \ekeys_cmd_add_args:e { { \tl_to_str:n { pt } } } }
    { \ekeys_cmd_add_args:n { { \BooleanTrue } } }
    \ekeys_cmd_scanner_end:
  }
  {
    \bool_if:NTF \l_tmpa_bool
    { \ekeys_cmd_add_args:n { { } } }
    { \ekeys_cmd_add_args:n { { \BooleanFalse } } }
    \ekeys_cmd_scanner_end:
  }
}
\ExplSyntaxOff
\ttfamily
\ekeysdeclarecmd \foo { &K{my/if-pt} m } {[#1](#2)}
\DeclareCommandCopy{\foocopied}{\foo}
```

```
\AddToHookWithArguments{cmd/foocopied/before}{\{#1|#2\}}
\foocopied pt; \foocopied PT; \foocopied p @; \quad
\foo pt; \foo PT; \foo p @;
```

```

{pt|;}[pt](;) {pt|;}[pt](;) {lp}[](p) @; [pt](;) [pt](;)
[](p) @;
```

例 90

```
\ExplSyntaxOn
\dim_new:N \l__my_scan_whd_dim % 临时保存长度
\seq_new:N \l__my_scan_whd_seq % 保存三个值, width,height,depth
\cs_generate_variant:Nn \seq_set_item:Nnn { NnV }
\collectn_set_keywords:Nn \c__my_scan_whd_tl { width, height, depth }
% 3 个参数, 为 width, height, depth
\keys_cmd_new_scanner:nnpn { my/scan-whd } { 3 }
{
  % initial seq
  \seq_clear:N \l__my_seq_whd_seq
  \seq_put_right:NV \l__my_seq_whd_seq \c_novalue_tl
  \seq_put_right:NV \l__my_seq_whd_seq \c_novalue_tl
  \seq_put_right:NV \l__my_seq_whd_seq \c_novalue_tl
  \__my_scan_whd_next:
}
\cs_new:Npn \__my_scan_whd_next:
{
  \collectn_scan_keywords:NTF \c__my_scan_whd_tl
  {
    \collectn_value:Nnw \l__my_scan_whd_dim
    {
      \seq_set_item:NnV \l__my_seq_whd_seq
        { \l_collectn_keywords_int } \l__my_scan_whd_dim
      \__my_scan_whd_next:
    } =
  }
  {
    \ekeys_cmd_add_args:e
    { \seq_map_function:NN \l__my_seq_whd_seq \ekeys_exp_not_braced:n
  }
  \ekeys_cmd_scanner_end:
}
}
\ExplSyntaxOff

\keysdeclarecmd \foo { K{my/scan-whd} }
{[ W: \IfValueTF{#1}{#1}{--}; H: \IfValueTF{#2}{#2}{--}; D: #3 ]}
\foo height 3pt depth \dimeval{3pt+2pt-10pt} height \dimexpr 3pt-2pt wide

[ W: --; H: 1.0pt; D: -5.0pt ]wide
```

TODO

- ☐ 实现宏包加载机制;
- ☐ 完善 `buffer` 及其文档;
- ☐ 更好的文档;
- ☐ 实现 `pgf` 库;
- ☐ 适配 `tcolorbox`;
- ☐ 提供 `cleveref` 宏包的接口;
- ☐ `cusclass` 文档类;
- ☐ 更好的竖排文档支持;
- ☐ `struct` 模块支持 `titlesec` 的所有标题形状;
- ☐ 更多丰富的标题设置;
- ☐ `struct` 模块支持 KOMA-Script 的目录设置方式;
- ☐ 优化现有的局部目录实现方式;
- ☐ `struct` 模块简单支持 `titletoc` 的目录设置方式;
- ☐ 适配 `latex-lab`;
- ☐ 文档部件 (`frontmatter`, `mainmatter`, `appendix` 等);
- ☐ 注记支持 (脚注、边注、尾注等);
- ☐ `colorful` 库, 彩色;
- ☐ `textdeco` 库, 文字装饰;
- ☐ 完善 `doc` 库;
- ☐ 子文档;
- ☐ 更加适配参考文献;
- ☐ 更加适配术语表;
- ☐ 实现 `hooks` 库;
- ☐ 实现 `external` 库;
- ☐ `tagged pdf`;
- ☐ 实现 `splitidx`;
- ☐ 更多测试;
- ☐ T_EX、L^AT_EX 内部命令的接口?
- ☐ `multicolrule`?
- ☐

索引

代码索引

粗体的数字表示描述对应索引项的页码；意大利体的数字表示使用对应索引项的页码。

Symbols	C
\+ 40	cbl-setup 的选项:
/tikz 的选项:	from 35
/tikz/at_tangent 71	to 35
/tikz/tangent 71	write 35
at_tangent 71	\chapter <i>29, 30, 32, 34, 56, 57</i>
tangent 71	clist 的命令:
	\clist_use:Nn <i>52</i>
A	\clist_use:nn <i>52</i>
\addcombinedlisttype 55, 60, 69	\clist_use:Nnnn <i>52</i>
\AddToHook 41	\clist_use:nnnn <i>52</i>
\alt 77	\cls 74
\Arg 74, 74	\cmd 72, 72, 73
\AtBeginDocument 84	cmd 的内部命令:
\atleastfiller 22	_cmd_peek_nonspace:NTF <i>122</i>
\automaticval 74	_cmd_peek_nonspace_remove:NTF <i>122</i>
	_cmd_token_if_cs:NTF <i>122</i>
B	collectn 的命令:
\background 27, 27, 28, 71	\collectn_box:Nnw 85, 85
\backgroundpicture 27	\collectn_delimited_all_inline:nn 88
\bigsize1 83	\collectn_delimited_any_inline:nn 88
\bigsizeM 83	\collectn_delimited_inline:NNn 87
\bigsizeR 83	\l_collectn_delimited_strip_bool 88
\BNFAnchor 77	\collectn_delimited_variable:NNNn 87
\BNFI 77, 77, 79	\collectn_gset_varwidth:Nnnw 85
\BNFItem 77, 77	\collectn_gset_varwidth_end: 85
\BNFN 77, 77, 79	\collectn_gset_vbox_varwidth:Nnw 85
\BNFO 77, 77, 79	\collectn_gset_vbox_varwidth_end: 85
\BNFref 77, 79	\collectn_gset_vbox_width:Nnw 85
\BNFT 77, 77, 79	\collectn_gset_vbox_width_end: 85
\bodybmargin 13	\collectn_hbox_auto:Nnnw 87, 87
\bodylmargin 13	\collectn_if_in:nTF 87
\bodyrmargin 13	\l_collectn_keywords_int 89, 90
\bodytmargin 13	\l_collectn_keywords_str 89, 90
\BooleanFalse 93, 94	\collectn_lohi:Nnw 88, 88
\BooleanTrue 93, 94	\collectn_lohis:Nnw 88, 88
bool 的命令:	\l_collectn_long_verbatim_bool 85, 85
\bool_if:NTF 38	\collectn_minipage:Nnnnnnw 80, 87
box 的命令:	\collectn_minipage:Nnnnw 86, 86
\box_use:N 55	\collectn_minipage:Nnnw 86
\breakablefiller 22	\collectn_minipage_auto:Nnnw 87
\breakablehspace 20	
\breakablevspace 20	

- \collectn_scan_keyword:n 89
- \collectn_scan_keyword:NTF 89
- \collectn_scan_keyword:nTF 89, 93
- \collectn_scan_keywordcs:n 89
- \collectn_scan_keywordcs:NTF 89
- \collectn_scan_keywordcs:nTF 89
- \collectn_scan_keywords:NTF 90
- \collectn_scan_keywords:nTF 89
- \collectn_scan_keywordscs:NTF 90
- \collectn_scan_keywordscs:nTF 89
- \collectn_set_keyword:Nn 89, 89
- \collectn_set_keywords:Nn 89, 90
- \collectn_set_keywords_from_seq:NN 89, 90
- \collectn_set_varwidth:Nnnw 85
- \collectn_set_varwidth_end: 85
- \collectn_set_vbox_varwidth:Nnw 85, 85
- \collectn_set_vbox_varwidth_end: 85
- \collectn_set_vbox_width:Nnw 85, 85
- \collectn_set_vbox_width_end: 85
- \collectn_value:Nnw 86
- \collectn_varwidth:Nnnnnnw 80, 87
- \collectn_varwidth:Nnnnw 86
- \collectn_varwidth:Nnnw 86
- \collectn_varwidth:nnw 85
- \collectn_varwidth_auto:Nnnw 87
- \collectn_varwidth_end: 85
- \collectn_verb:Nnw 85
- \collectn_width:Nnnnnnw 87
- \collectn_width:Nnnnw 86, 86
- \collectn_width:Nnnw 86
- \collectn_width_auto:Nnnw 87
- 颜色名称:
 - cus/color/doc_cs 76
 - cus/color/doc_env 76
 - cus/color/doc_key 76
 - cusfiller 20
- color 的命令:
 - \color_select:n 20, 45
- \coppagestyle 15, 15
- \cs 72, 72, 73
- \csref 76, 76
- \csreflist 76
- \csstring 2
- cs 的命令:
 - \cs_generate_from_arg_count:NNnn 91
 - \cs_set_protected:Npn 91
 - \cs_to_str:N 75
- \CurrentCombinedListCount 57, 68
- \CurrentTitleLevel 31
- \CurrentTitleName 31
- \CurrentTocDefaultLevelCount 57
- \CUSDependency 41
- \cusemoji 4
- \cusemojilowerratio 4, 4
- \cusemojitotalratio 4, 4
- \CusLaTeX 1
- \CUSLibraryDelayedUntil 41
- \CUSLoadLibrary 41
- \CUSProvideExplLibrary 41
- \CUSProvideLibrary 41
- \cussetstyle 2
- \cussetup 2, 2, 36, 37
- \CusTeX 1
- cus 的命令:
 - \cus_act_case:nnnn 52, 52
 - \cus_act_case_false:nnnn 52
 - \cus_act_case_true:nnnn 52, 52
 - \g_cus_anchor_tl 44, 44
 - \cus_arg_to_keyval_apply:nnN 47
 - \cus_arg_to_keyval_apply:nnn 47
 - \cus_bookmark:nn 45, 45
 - \cus_contents_get:nN 58
 - \cus_contents_type_get:nnN 58
 - \g_cus_doc_module_alias_prop 76
 - \g_cus_doc_type_alias_prop 76
 - \cus_exp_after_num:nwN 52
 - \cus_exp_args:Nd 50
 - \cus_exp_args:NNd 50
 - \cus_exp_args:Nnd 50
 - \cus_exp_args:nw 51, 51
 - \cus_exp_last_unbraced:Nd 50
 - \cus_exp_last_unbraced:NNd 50
 - \cus_exp_last_unbraced:Nnd 50
 - \cus_exp_num:nN 52
 - \cus_get_graphics_full_name:nN 43
 - \cus_get_graphics_full_name:nNTF 43
 - \cus_get_heading_level:nnN 58
 - \cus_gset_next_anchor_name:n 44, 44
 - \cus_gset_next_anchor_raise:n 44, 44
 - \cus_gset_next_bookmark_text:n 45
 - \cus_gset_next_page_label:n 46, 46
 - \cus_gset_page_label:n 46
 - \cus_gset_page_label_code:n 46
 - \cus_hyper_anchor:n 45
 - \cus_hyper_anchor_start:n 45
 - \cus_hyper_anchor_stop: 45
 - \cus_hyper_link:nnn 45
 - \cus_hyper_link_file:nnn 45

<code>\cus_hyper_link_launch:nnn</code>	45	<code>\cus_split_bracket_head_default:nn</code>	48
<code>\cus_hyper_link_name:nn</code>	45	<code>\cus_split_bracket_tail:n</code>	48
<code>\cus_hyper_link_start:nn</code>	45	<code>\cus_split_bracket_tail_default:nn</code>	48
<code>\cus_hyper_link_stop:</code>	45	<code>\cus_tl_split_braced:nn</code>	49, 49
<code>\cus_hyper_link_url:nn</code>	45	<code>\cus_tl_split_braced:NNNN</code>	50, 50
<code>\cus_if_after_documentclass:TF</code>	50	<code>\cus_tl_use:Nn</code>	52
<code>\cus_if_after_documentclass_p:</code>	50	<code>\cus_tl_use:nn</code>	52
<code>\cus_if_class_loaded:TF</code>	50	<code>\cus_tl_use:Nnnn</code>	52
<code>\cus_if_class_loaded_p:</code>	50	<code>\cus_tl_use:nnnn</code>	52
<code>\cus_if_document:TF</code>	50	<code>\cus_use_none_num:nw</code>	51
<code>\cus_if_document_p:</code>	50	<code>\cus_use_shbox:N</code>	55, 55
<code>\cus_if_head_int:nTF</code>	46	<code>\CUSDependency</code>	41
<code>\cus_if_head_int_p:n</code>	46	<code>\CUSLibraryDelayedUntil</code>	41
<code>\cus_if_keyval_apply:NNN</code>	47	<code>\CUSLoadLibrary</code>	41
<code>\cus_if_keyval_apply:nnn</code>	47	<code>\CUSPassOptionsToLibrary</code>	41
<code>\cus_if_keyval_variable:nNnn</code>	47	<code>\CUSProvideExplLibrary</code>	40
<code>\cus_if_lazy_all:nnTF</code>	53	<code>\CUSProvideLibrary</code>	40
<code>\cus_if_lazy_any:nnTF</code>	53		
<code>\cus_if_pdfstring:TF</code>	45		
<code>\cus_if_pdfstring_p:</code>	45		
<code>\cus_if_preamble:TF</code>	50		
<code>\cus_if_preamble_p:</code>	50		
<code>\cus_label_info:nn</code>	44		
<code>\cus_make_target:n</code>	44, 44		
<code>\cus_make_unique_target:n</code>	44, 44		
<code>\cus_map_nest_code:Nnnn</code>	53		
<code>\cus_map_nest_variable:NnnNn</code>	53		
<code>\cus_newlabel_now:nnnnnn</code>	44		
<code>\cus_newlabel_shipout:nnnnnn</code>	44		
<code>\cus_newlabel_shipout_x:nnnnnn</code>	44		
<code>\cus_peek_act:nnnnn</code>	46		
<code>\cus_peek_shbox:Nnw</code>	55		
<code>\cus_peek_verbatim:nw</code>	46		
<code>\cus_ref_label:nn</code>	45		
<code>\cus_ref_label_box:nn</code>	45, 54, 55, 79		
<code>\cus_ref_label_shbox:nn</code>	45, 55, 79		
<code>\cus_ref_label_varwidth:nnnn</code>	54		
<code>\cus_ref_label_width:nnnn</code>	54		
<code>\cus_ref_target:nn</code>	45		
<code>\cus_ref_target_box:nn</code>	45, 54, 55, 79		
<code>\cus_ref_target_shbox:nn</code>	45, 55, 79		
<code>\cus_ref_target_varwidth:nnnn</code>	54		
<code>\cus_ref_target_width:nnnn</code>	54		
<code>\cus_reset_page_label_code:</code>	46		
<code>\cus_serial_exp_args:nw</code>	51		
<code>\cus_set_shbox:Nn</code>	55		
<code>\cus_set_shbox:Nw</code>	55, 55		
<code>\cus_set_shbox_end:</code>	55		
<code>\cus_split_bracket_head:n</code>	48		
		D	
		<code>\dashfiller</code>	19, 21
		<code>\DeclareDocumentCommand</code>	90, 91
		<code>\DeclareEKeysCollector</code>	91
		<code>\DeclareEKeysCommand</code>	91, 91
		<code>\DeclareInstanceCopy</code>	61
		<code>\definetitle</code>	29
		<code>\delsizel</code>	83
		<code>\delsizem</code>	83
		<code>\delsizer</code>	83
		dependency 的选项:	
		disable	41
		library	41
		module	41
		package	41
		<code>\dimeval</code>	10, 13
		doc/cmd 的选项:	
		break-at-any	73
		doc/cmd/type	76
		hyper	73
		hyphen-opacity	73
		index	73
		keyval	73
		module	73
		no-index	73
		space	73
		target	73
		type	73
		doc/function 的选项:	
		added	75
		EXP	75
		frame	76

- frame+ 76
 - label 75
 - label* 75
 - module 75
 - no-label 75
 - noTF 75
 - path 75
 - pTF 75
 - rEXP 75
 - TF 75
 - type 75
 - updated 75
 - verb 75
 - \docfile 74
 - \DocumentMetadata 7, 45, 84
- E**
- \ekeyscollectargs 92, 92
 - \ekeyscollectargs* 92
 - \ekeyscollectorarg 91
 - \ekeysdeclarecmd 90, 91, 91, 92, 95, 96
 - \ekeysdeclarecollector 91
 - \EKeysEndPreprocessor 99, 99
 - \ekeysgenerategrabber 92, 92
 - \ekeysmakeglobal 92
 - \ekeysnewscanneralias 98, 98
- ekeys 的内部命令:
- __ekeys_grabber_args_do:Nnn 96
 - __ekeys_if_cmd:NTF 93
- ekeys 的命令:
- \ekeys_cmd_add_arg:n 100
 - \ekeys_cmd_add_args:n 98, 100
 - \ekeys_cmd_after_declare:n 95, 97
 - \ekeys_cmd_args: 98, 100
 - \l_ekeys_cmd_defaults_bool 91, 91
 - \ekeys_cmd_global:N 92
 - \l_ekeys_cmd_max_args_int 92
 - l_ekeys_cmd_max_args_int 92
 - \ekeys_cmd_name: 94, 98, 100
 - \ekeys_cmd_new_scanner:nnnpn 97
 - \ekeys_cmd_new_scanner:nnnpn 97
 - \ekeys_cmd_new_scanner:nnpn 97
 - \ekeys_cmd_new_scanner:nnpx 97
 - \ekeys_cmd_new_scanner_alias:nn 98
 - \ekeys_cmd_new_scanner_alias:nnn 98
 - \ekeys_cmd_register_cs:N 100
 - \ekeys_cmd_scanner_alias_undefine:n 100
 - \l_ekeys_cmd_scanner_args_int 99, 99
 - \ekeys_cmd_scanner_end: 98, 99, 99
 - \ekeys_cmd_scanner_undefine:n 100
 - \ekeys_cmd_undefine:N 92, 100
 - \ekeys_collect_args_do:Nnn 92
 - \ekeys_collect_args_do:Nnn 92
 - \ekeys_declare_cmd:Nnn 91, 91
 - \ekeys_declare_cmd_x:Nnn 91, 91
 - \ekeys_exp_not_braced:n 95
 - \ekeys_grabber_args_do:Nnn 92, 96
- elkernel 的命令:
- \elkernel_arg_to_keyvalue:Nnn 90
 - \elkernel_if_date_at_least:nnTF 90
- \enablecombinedlist 35, 35, 56, 59
 - \endminipage 122
 - \ensurefourdigits 83
 - \ensurethreedigits 83
 - \ensuretwdigits 83
 - enumlist 4
 - enumlist* 4
 - \env 74
 - \envref 76, 76
 - \envreflist 76
 - \ExpandArgs 6, 51
 - \ExpandArgument 6, 7
- exp 的命令:
- \exp_not:n 49, 52, 95
- F**
- \fancycenter 16
 - \file 74
 - \filler 19, 38
- filler 的选项:
- align 21
 - box 20
 - box* 20
 - cdotted 21
 - center 21
 - clear-box 20
 - color 20
 - content 20
 - dash 21
 - dashed 21
 - dotted 21
 - filler/size 20
 - filler/size* 20
 - full 21
 - hspace 20
 - hspace* 20
 - is-dash 21
 - not-space 20
 - raise 21
 - rule 21

sep	21	\hi	3, 3, 77
size	19	内部钩子:	
size*	19	_hyp/target/setname	119
solid	21	钩子:	
space	20	begindocument	84
spread	21	begindocument/end	84
type	21	insertmark	58, 58
vspace	20	label	44
vspace*	20	shipout	9
\FirstMark	59	shipout/foreground	9
\FloatBarrier	33, 33	shipout/after	9
\forbiddenval	74	shipout/background	9
\foreground	27, 27, 71	shipout/before	46
\foregroundpicture	27	hook 的内部命令:	
\fparbox	80, 83, 83	_hook_patch_expand_redefine:Nnnn ...	122
frame/outer-sep	2	hook 的命令:	
frame/sep	2	\AddToHook	41
Framed	18	\AddToHookNext	42
frame 的选项:		\AddToHookNextWithArguments	42
align	18	\AddToHookWithArguments	41
center	18	\ClearHookNext	42
first	18	\g_hook_patch_action_list_tl	122
first*	18	\RemoveFromHook	42
frame	18	\UseHook	41
frame*	18	\UseHookWithArguments	41
ignore-warnings	19	\HyperLink	54, 62, 79, 79, 80
init	18	\HyperRef	54, 79, 79, 80, 83
inner	18	hyperref 的内部命令:	
last	18	_hyp_target_counter_anon:n	119
last*	18	\l_hyp_target_create_bool	119
left	18	_hyp_target_manual:nn	119
middle	18	I	
middle*	18	\IfAbsPageOdd	9, 9, 79
outer	18	\ifbotfloat	16
outer-sep	18	\iffloatpage	16
ratio	18	\iffootnote	16
right	18	\IfGraphicsExists	4
rule-width	18	\ifincblentry	34, 34
sep	18	\ifintitle	34
whole	18	\IfIsCounterTF	84
whole*	18	\IfLabelOdd	79
width	18	\IfNoValueTF	44
function	74	\IfPageOdd	9, 79
\fvarbox	80, 83, 83	\IfQuarkNoValueF	94
G		\IfQuarkNoValueT	94
\getcbllevelrange	36, 58	\IfQuarkNoValueTF	91, 94
H		\IfRecordTotalCounterTF	84
hbox 的命令:		\iftopfloat	16
\hbox:n	85	if 的命令:	
		\if_predicate:w	75

<code>\includegraphicspx</code>	84, 84	<code>\layoutloffset</code>	10
<code>\index</code>	29, 29	<code>\layoutroffset</code>	10
<code>\initemptyval</code>	74	<code>\layouttoffset</code>	10
<code>\initialval</code>	74	<code>\layoutwidth</code>	10
<code>\InputIfGraphicsExists</code>	4	layout 的选项:	
<code>\InsertMark</code>	58	<code>asymmetric</code>	13
<code>\is</code>	77	<code>bindingoffset</code>	13
<code>\IterateClist</code>	8, 8	<code>bmarg</code>	13
<code>\iteratecontents</code>	57	<code>body</code>	11
<code>\IterateInteger</code>	8, 8	<code>bottom</code>	13
<code>\IterateList</code>	8, 8	<code>bottommargin</code>	13
<code>\IterateThread</code>	8	<code>centering</code>	13
K			
内核保留的命令:			
<code>__kernel_backend_literal_pdf:n</code>	122	<code>centerlayout</code>	10
<code>__kernel_chk_if_free_cs:N</code>	122	<code>columnsep</code>	14
<code>__kernel_cmd_if_xparse:NTF</code>	122	<code>direction</code>	10
<code>__kernel_file_name_sanitiz</code>	122	<code>divide</code>	12
<code>__kernel_quark_new_conditional:Nn</code>	122	<code>foot</code>	14
<code>__kernel_quark_new_test:N</code>	122	<code>footnotesep</code>	13
<code>__kernel_str_to_other_fase:n</code>	122	<code>footoffset</code>	15
<code>\key</code>	72	<code>footskip</code>	14
<code>\keyref</code>	76, 76	<code>hcentering</code>	13
<code>\keyreflist</code>	76	<code>hdivide</code>	12
keys 的内部命令:		<code>head</code>	14
<code>\c__keys_code_root_str</code>	122	<code>headheight</code>	14
<code>\c__keys_inherit_root_str</code>	122	<code>headoffset</code>	15
<code>\l__keys_inherit_str</code>	122	<code>headsep</code>	14
<code>\l__keys_module_str</code>	122	<code>height</code>	11
<code>\c__keys_props_root_str</code>	122	<code>heightrounded</code>	12
<code>\c__keys_type_root_str</code>	122	<code>hoffset</code>	15
keys 的命令:		<code>hmargin</code>	13
<code>.clist_put_left:N</code>	119	<code>hmarginratio</code>	13
<code>.clist_put_right:N</code>	119	<code>hoffset</code>	14
<code>.gbey_psrrule:nn</code>	119	<code>horizontalmargin</code>	13
<code>\l_keys_path_tl</code>	122	<code>horizontalmarginratio</code>	13
<code>\keys_precompile:nnN</code>	40	<code>hscale</code>	10
<code>\keys_set:nn</code>	40	<code>ignoreall</code>	12
<code>.obey_psrrule:nn</code>	119	<code>ignorefoot</code>	12
<code>.switch_set:N</code>	119	<code>ignorehead</code>	12
<code>.toks_put_left:N</code>	119	<code>ignoreheadfoot</code>	12
<code>.toks_put_right:N</code>	119	<code>ignorehf</code>	12
keyval	74	<code>ignoremarginpar</code>	12
L			
<code>\labelinfo</code>	44	<code>ignoremp</code>	12
<code>\LastMark</code>	59	<code>includeall</code>	12
<code>latexbnf</code>	77	<code>includefoot</code>	11
<code>\layoutboffset</code>	10	<code>includehead</code>	11
<code>\layoutheight</code>	10	<code>includeheadfoot</code>	11
		<code>includehf</code>	11
		<code>includemarginpar</code>	11
		<code>includemp</code>	11

- `\marg` 74, 74
mark 的内部命令:
`\g__mark_..._tl` 122
`\c__mark_class_..._mark` 122
`\g__mark_classes_seq` 122
`__mark_update_structure_alias:nn` ... 122
mark 的命令:
`\FirstMark` 59
`\IfMarksEqualTF` 59
`\InsertMark` 58
`\LastMark` 59
`\mark_if_eq:nnnnnnTF` 59
`\mark_if_eq:nnnnTF` 59
`\mark_insert:nn` 58, 58
`\mark_new_class:n` 58
`\mark_use_first:nn` 59
`\mark_use_last:nn` 59
`\mark_use_top:nn` 59
`\NewMarkClass` 58
`\TopMark` 59
`\meta` 74, 74
`\multicollocalplaincombinedlist` 36
`\multicolplaincombinedlist` 36, 36, 60
multicolumns 的选项:
`addto-baselineskip` 25
`adj` 25
`adj*` 25
`adj-inner` 25
`adj-outer` 25
`aligned` 24
`balanced` 24
`bottom-fuzz` 25
`collectmore` 25
`cols` 23
`cols*` 24
`column-badness` 25
`column-sep` 23
`columns` 23
`columns*` 24
`disable-swap-column` 24
`enable-swap-column` 24
`final-column-badness` 25
`first-minimal` 23
`flush` 24
`framed` 24
`framed-options` 24
`framed-options+` 24
`h-fuzz` 25
`heading` 23
`last-minimal` 23
`left-to-right` 26
`LR` 26
`minrows` 25
`multicolumns/adj` 23
`multicolumns/framed-options` 72, 83
`multicolumns/not-balanced` 25
`multicolumns/overflow` 24
`not-aligned` 24
`not-balanced` 24
`outer-sep` 23
`overflow` 25
`pretolerance` 25
`ragged` 24
`right-to-left` 26
`RL` 26
`rule-color` 23
`rule-width` 23
`sep` 23
`shorten` 25
`swap-column` 24
`tolerance` 25
`top-fuzz` 25
`unbalance` 25
`v-fuzz` 25
`widen` 25
`wrap-box` 25

N

`\NewCommandCopy` 91
`\NewDocumentCommand` 5
`\newindextype` 28, 28
`\normalcolseprulecolor` 82
`\normalcolumncolor` 82
novalue 的命令:
`\c_novalue_tl` 44
`\NumberCase` 94
`\numberfixedwidth` 3
`\numberzerofill` 3
`\numerzerofill` 3

O

`\oarg` 74, 74
`\opt` 74
`\orbar` 75

P

paracol 的命令:
`\colseprulecolor` 82
`\columncolor` 82
`\columnwidth` 81

[illegible]

S	
<code>\SaveSpecifiedCombinedListStyle</code>	40, 40
scan 的内部命令:	
<code>\s_seq</code>	119
<code>\s__tl_stop</code>	119
<code>\section</code>	29, 30, 31, 33, 34
seq 的内部命令:	
<code>_seq_item:n</code>	119
<code>_seq_pop_item_def:</code>	119
<code>_seq_push_item_def:n</code>	119
seq 的命令:	
<code>\seq_map_function:NN</code>	75
<code>\seq_map_tokens:Nn</code>	53
<code>\setcenterfoot</code>	16
<code>\setcenterhead</code>	16
<code>\setfoot</code>	16, 16
<code>\setfootinit</code>	16
<code>\setfootrule</code>	16
<code>\setfootruleskip</code>	16
<code>\setfootrulewidth</code>	16
<code>\setgraphicspath</code>	4
<code>\sethead</code>	16, 16
<code>\setheadfoot</code>	16
<code>\setheadfootinit</code>	16
<code>\setheadinit</code>	16
<code>\setheadrule</code>	16
<code>\setheadruleskip</code>	16
<code>\setheadrulewidth</code>	16
<code>\setindexinit</code>	29
<code>\setindexprologue</code>	29
<code>\setinputpath</code>	4
<code>\setleftfoot</code>	16
<code>\setleftthead</code>	16
<code>\setpagestyle</code>	15, 16
<code>\setrightfoot</code>	16
<code>\setrightthead</code>	16
<code>\setsecnumdepth</code>	31, 31
<code>\SetSpecifiedCombinedListStyle</code> ..	40, 40, 61–63
<code>\settocdepth</code>	35, 36, 60
<code>\setupindex</code>	28, 28
<code>\setuplayout</code>	9
<code>\setupnexttitle</code>	30
<code>\setuponetime</code>	30
<code>\setuptitle</code>	30, 30
<code>\shadecontent</code>	70
<code>\shadecontentbox</code>	70
<code>\shadetext</code>	70
<code>\shadetextbox</code>	70
shipout 的命令:	
<code>\g_shipout_readonly_int</code>	46
<code>\g_shipout_totalpages_int</code>	46
<code>\ShowCommand</code>	91
<code>\specified</code>	39
<code>\SpecifiedCombinedList</code>	40, 61, 62
<code>\specifiedlof</code>	39, 61
<code>\specifiedlot</code>	39, 61
<code>\specifiedtoc</code>	39
<code>\SplitArgument</code>	5
<code>\SplitList</code>	5
<code>\spreadtext</code>	17, 17
<code>\startextrawidth</code>	26
<code>\startfullpagewidth</code>	26
<code>\startmulticolumns</code>	23, 63
<code>\startparacol</code>	80
<code>\startrotate</code>	26
<code>\stopextrawidth</code>	26
<code>\stopfullpagewidth</code>	26
<code>\stopmulticolumns</code>	23
<code>\stopparacol</code>	80
<code>\stoprotate</code>	26
<code>\subparagraph</code>	30
<code>\subsection</code>	30
<code>\subsubsection</code>	30
<code>syntax</code>	74
T	
<code>\tableofcontents</code>	35, 61
templatecbl 的选项:	
<code>*</code>	36
<code>hide</code>	36
<code>no-parent</code>	36
<code>show</code>	36
<code>\templatecombinedlist</code>	36
<code>\templatelof</code>	36
<code>\templatelot</code>	36
<code>\templatetoc</code>	36, 36
T _E X 和 L ^A T _E X 2 _ε 的命令:	
<code>\-</code>	7
<code>\@@vwid@minipagerestore</code>	120
<code>\@DeclareFloatingEnvironment</code>	120
<code>\@acci</code>	120
<code>\@accii</code>	120
<code>\@acciii</code>	120
<code>\@afterheading</code>	121
<code>\@auxout</code>	119, 120
<code>\@bsphack</code>	120
<code>\@citecolor</code>	119
<code>\@curr@file</code>	4
<code>\@currentHref</code>	44, 119
<code>\@currentlabel</code>	119

<code>\@currpagestyle</code>	15	<code>\arabic</code>	83
<code>\@declarecommandcopylisthook</code>	122	<code>\balance@columns</code>	122
<code>\@dischyp</code>	7, 119, 120	<code>\balance@columns@out</code>	122
<code>\@elt</code>	121	<code>\baselineskip</code>	12
<code>\@filecolor</code>	119	<code>\bBigg@</code>	121
<code>\@idxitem</code>	120	<code>\bigl</code>	83
<code>\@ifabspageodd</code>	9	<code>\bigm</code>	83
<code>\@ifpageodd</code>	9	<code>\bigr</code>	83
<code>\@iiiparbox</code>	120	<code>\BKM@hook</code>	119
<code>\@indexfile</code>	120	<code>\bookmark</code>	45
<code>\@input</code>	120	<code>\bookmark@text</code>	119
<code>\@input@</code>	120	<code>\c@columnbadness</code>	120
<code>\@kernel@after@enddocument@afterlastpage</code>	120	<code>\c@finalcolumnbadness</code>	120
<code>\@linkcolor</code>	119	<code>\c@minrows</code>	120
<code>\@makestophead</code>	121	<code>\c@page</code>	119–121
<code>\@maketophead</code>	121	<code>\c@secnumdepth</code>	121
<code>\@minipagerestore</code>	120	<code>\c@tocdepth</code>	121
<code>\@mpfn</code>	120	<code>\c@unbalance</code>	120
<code>\@nameuse</code>	56	<code>\cdot</code>	21
<code>\@notprerr</code>	119	<code>\chapter</code>	36, 69, 121
<code>\@numbercase</code>	94	<code>\chaptermark</code>	34
<code>\@onlypreamble</code>	119	<code>\cl@ckpt</code>	84, 121
<code>\@outputpage</code>	119	<code>\cleaders</code>	21
<code>\@parboxrestore</code>	120	<code>\color</code>	20, 45
<code>\@rotdblfloat</code>	120	<code>\color@hbox</code>	120
<code>\@rotfloat</code>	120	<code>\colorlet</code>	76
<code>\@sanitize</code>	120	<code>\columnratio</code>	81
<code>\@secpenalty</code>	121	<code>\columnsep</code>	14, 23, 81
<code>\@setminipage</code>	120	<code>\columnseprule</code>	23
<code>\@showcommandlisthook</code>	122	<code>\columnseprulecolor</code>	23
<code>\@specialstyle</code>	15, 120	<code>\contentsline</code>	121
<code>\@starttoc</code>	121	<code>\contentsname</code>	35, 36
<code>\@stpelt</code>	121	<code>\CTEX@addloflotskip</code>	121
<code>\@svsec</code>	121	<code>\cus@cbl@contentsline</code>	56
<code>\@svsechd</code>	121	<code>\cus@cbl@contentsline@</code>	56
<code>\@tempdimd</code>	121	<code>\cus@colorcite</code>	45
<code>\@topnewpage</code>	121	<code>\cus@colorfile</code>	45
<code>\@totalleftmargin</code>	120	<code>\cus@colorlink</code>	45
<code>\@twoclasseserror</code>	119	<code>\cus@colormenu</code>	45
<code>\@urlcolor</code>	119	<code>\cus@colorrun</code>	45
<code>\@vwid@setup</code>	120	<code>\cus@colorurl</code>	45
<code>\@wrindex</code>	120	<code>\cus@doc@cls@format</code>	74
<code>\@writefile</code>	120	<code>\cus@doc@cmd@format</code>	72
<code>\@xnthm</code>	120	<code>\cus@doc@cs@format</code>	72, 72
<code>\addcontentsline</code>	56	<code>\cus@doc@csref@format</code>	76, 76
<code>\addtocontents</code>	121	<code>\cus@doc@env@format</code>	74
<code>\adjmc@process@ne@column</code>	122	<code>\cus@doc@envref@format</code>	76
<code>\allocationnumber</code>	121	<code>\cus@doc@itfont</code>	76
		<code>\cus@doc@key@format</code>	72

<code>\cus@doc@keyref@format</code>	76	<code>\FB@captype</code>	120
<code>\cus@doc@marg@format</code>	74	<code>\fbox</code>	24
<code>\cus@doc@meta@format</code>	74, 74	<code>\fcolorbox</code>	70
<code>\cus@doc@oarg@format</code>	74	<code>\file</code>	74
<code>\cus@doc@opt@format</code>	74	<code>\FloatBarrier</code>	121
<code>\cus@doc@parg@format</code>	74	<code>\floatplacement</code>	120
<code>\cus@doc@pkg@format</code>	74	<code>\flushcolumns</code>	24
<code>\cus@doc@refrange</code>	76	<code>\footins</code>	13
<code>\cus@doc@tn@format</code>	72	<code>\footruleskip</code>	16
<code>\cus@doc@ttfont</code>	76	<code>\footrulewidth</code>	16
<code>\cus@doc@veta@format</code>	74	<code>\footskip</code>	11, 14
<code>\cus@filename</code>	43	<code>\frozen@everydisplay</code>	121
<code>\cus@getgraphicsname</code>	43	<code>\frozen@everymath</code>	121
<code>\cus@type@contentsline</code>	56	<code>\Gin@ii</code>	122
<code>\cus@type@contentsline@</code>	56	<code>\Gin@include@graphics</code>	119
<code>\cusframerule</code>	18	<code>\Gin@input@path</code>	119
<code>\cusframesep</code>	18	<code>\globalcounter</code>	82
<code>\DeclareNewFloatType</code>	120	<code>\Gm@changelayout</code>	119
<code>\declaretheorem</code>	60	<code>\Gm@layoutheight</code>	120
<code>\def</code>	46, 98	<code>\Gm@layoutwidth</code>	120
<code>\detokenize</code>	6	<code>\Gm@restore</code>	119
<code>\dimexpr</code>	10, 13	<code>\Gm@save</code>	119
<code>\docfile</code>	74	<code>\Gm@setsize</code>	119
<code>\documentclass</code>	50	<code>\Gm@warning</code>	119
<code>\dotfill</code>	19	<code>\Grot@setangle</code>	120
<code>\e@alloc@chardef</code>	121, 122	<code>\H@refstepcounter</code>	121
<code>\e@alloc@top</code>	92, 122	<code>\hbox</code>	45, 54, 55, 85
<code>\empty</code>	28	<code>\headheight</code>	11, 14
<code>\end@rotdblfloat</code>	120	<code>\headruleskip</code>	16
<code>\end@rotfloat</code>	120	<code>\headrulewidth</code>	16
<code>\endmulticols</code>	122	<code>\headsep</code>	11, 14
<code>\endtcbox@lrbox</code>	122	<code>\hfuzz</code>	25
<code>\endtcbox@savebox</code>	122	<code>\hoffset</code>	14
<code>\escapechar</code>	95	<code>\hrulefill</code>	19
<code>\etocsetstyle</code>	39, 61	<code>\hspace</code>	19, 20, 56
<code>\f@nch@def</code>	120	<code>\Hy@linktoc</code>	120
<code>\f@nch@footinit</code>	120	<code>\Hy@raisedlink</code>	44
<code>\f@nch@headinit</code>	120	<code>\Hy@RestoreSpaceFactor</code>	119
<code>\f@nch@O@elf</code>	119, 121	<code>\Hy@SaveSpaceFactor</code>	119
<code>\f@nch@O@elh</code>	119, 121	<code>\Hy@tocdestname</code>	120
<code>\f@nch@O@erf</code>	119, 121	<code>\hyper@anchor</code>	119
<code>\f@nch@O@erh</code>	119, 121	<code>\hyper@anchorend</code>	119
<code>\f@nch@O@olf</code>	119, 121	<code>\hyper@anchorstart</code>	119
<code>\f@nch@O@olh</code>	119, 121	<code>\hyper@link</code>	119, 120
<code>\f@nch@O@orf</code>	119, 121	<code>\hyper@linkend</code>	119
<code>\f@nch@O@orh</code>	119, 121	<code>\hyper@linkfile</code>	119
<code>\f@nch@pagestyle</code>	120	<code>\hyper@linklaunch</code>	119
<code>\f@nch@ps@<pagestyle>-is-fancyhdr</code>	120	<code>\hyper@linknamed</code>	119
<code>\f@size</code>	121	<code>\hyper@linkstart</code>	119

<code>\hyper@linkurl</code>	119	<code>\linewidth</code>	19
<code>\hyper@nopatch@sectioning</code>	121	<code>\listfigurename</code>	35, 36
<code>\HyperDestNameFilter</code>	44	<code>\listoffigures</code>	55, 60
<code>\hyperget</code>	44	<code>\listoftables</code>	55, 60
<code>\hyperlink</code>	56, 62, 79	<code>\listoftheorems</code>	120
<code>\hyperref</code>	79	<code>\listtablename</code>	35, 36
<code>\HyPL@EveryPage</code>	119	<code>\localcounter</code>	82
<code>\HyPL@page</code>	119	<code>\long</code>	91
<code>\HyPL@thisLabel</code>	119	<code>\LR@column@boxes</code>	120
<code>\if@fcolmade</code>	120	<code>\LRmulticolcolumns</code>	26
<code>\if@in@minipage@env</code>	120	<code>\makebox</code>	3, 87, 96
<code>\if@nobreak</code>	120	<code>\makeindex</code>	120
<code>\if@noskipsec</code>	120	<code>\marginparsep</code>	11
<code>\if@restonecol</code>	121	<code>\marginparwidth</code>	11
<code>\iff@nch@footnote</code>	120	<code>\markboth</code>	59
<code>\ifG@refundefined</code>	121	<code>\markright</code>	59
<code>\IfGraphicsExists</code>	119	<code>\maxbalancingoverflow</code>	25
<code>\ifHy@pdfstring</code>	119	<code>\mc@align@columns</code>	120
<code>\ifodd</code>	38	<code>\mult@nat@firstbox</code>	120
<code>\ifpcol@bg@swap</code>	121	<code>\mult@rightbox</code>	120
<code>\ifpcol@swapcolumn</code>	121	<code>\multi@column@out</code>	120, 122
<code>\ifpcol@swapmarginpar</code>	121	<code>\multicolbaselineskip</code>	25
<code>\ifthmt@isstarred</code>	120	<code>\multicolovershoot</code>	25
<code>\ifthmt@listswap</code>	120	<code>\multicolpretolerance</code>	25
<code>\ifx</code>	59	<code>\multicolsep</code>	23
<code>\includegraphics</code>	85	<code>\multicoltolerance</code>	25
<code>\includegraphicspax</code>	85	<code>\multicolundershoot</code>	25
<code>\index</code>	120	<code>\newcounter</code>	84
<code>\indexname</code>	28	<code>\newfloat</code>	120
<code>\input@path</code>	119	<code>\newfloat@setoptions</code>	120
<code>\InputIfGraphicsExists</code>	119	<code>\NEWPAX@AddAnnots</code>	122
<code>\is@specialpage</code>	120	<code>\NEWPAX@cmd@<type></code>	122
<code>\jobname</code>	28, 35	<code>\NEWPAX@file</code>	122
<code>\kvtcb@bottom</code>	122	<code>\NEWPAX@Gin@page</code>	122
<code>\kvtcb@bottom@rule@stand</code>	122	<code>\NEWPAX@includegraphics</code>	122
<code>\kvtcb@boxsep</code>	122	<code>\NEWPAX@skip</code>	122
<code>\kvtcb@left@rule@stand</code>	122	<code>\newtheorem</code>	60
<code>\kvtcb@leftupper</code>	122	<code>\nolinkurl</code>	74
<code>\kvtcb@new@listof</code>	120	<code>\nonumberline</code>	56
<code>\kvtcb@new@listtype</code>	120	<code>\normalbaselineskip</code>	44
<code>\kvtcb@right@rule@stand</code>	122	<code>\NR@getttitle</code>	121
<code>\kvtcb@rightupper</code>	122	<code>\NR@nopatch@sectioning</code>	121
<code>\kvtcb@top</code>	122	<code>\null</code>	77
<code>\kvtcb@top@rule@stand</code>	122	<code>\numberline</code>	56
<code>\l@<name></code>	69	<code>\outer</code>	46
<code>\label</code>	43, 75, 79	<code>\page@sofar</code>	120
<code>\leaders</code>	21	<code>\pagestyle</code>	119
<code>\leftmark</code>	59	<code>\par</code>	39, 85
<code>\lfbox</code>	83	<code>\paragraph</code>	69, 121

<code>\parbox</code>	3, 79, 80, 87, 97	<code>\tcbox@drawing@env@end</code>	122
<code>\part</code>	69, 121	<code>\tcbox@proc@options@init</code>	120
<code>\pcol@columnratioleft</code>	121	<code>\tcbox@split@L</code>	122
<code>\pcol@columnratoright</code>	121	<code>\tcbox@split@SL@displayed</code>	122
<code>\pcol@colwidthspecleft</code>	121	<code>\tcbox@split@start</code>	122
<code>\pcol@colwidthspecright</code>	121	<code>\tcbox@split@USL</code>	122
<code>\pcol@gcounters</code>	121	<code>\tcbox@vsplit@lower</code>	122
<code>\pcol@globalcounter</code>	121	<code>\tcbox@vsplit@upper</code>	122
<code>\pcol@mpthreshold@l</code>	121	<code>\tcbox</code>	70, 72
<code>\pcol@mpthreshold@r</code>	121	<code>\tcbox@inner@box</code>	122
<code>\pdfsavepos</code>	33	<code>\textheight</code>	11
<code>\pgf@declareimage</code>	121	<code>\textsubscript</code>	3
<code>\pgfdeclareimage</code>	70	<code>\textsuperscript</code>	3
<code>\pgfimage</code>	70	<code>\textvisiblespace</code>	73
<code>\postmulticols</code>	23	<code>\textwidth</code>	11, 81
<code>\premulticols</code>	23	<code>\tf@toc</code>	120
<code>\prep@keptmarks</code>	122	<code>\the</code>	10, 13
<code>\prepare@multicols</code>	120	<code>\the<counter></code>	82
<code>\printindex</code>	120	<code>\thepage</code>	39, 44, 46, 56
<code>\protected</code>	91	<code>\thispagestyle</code>	119
<code>\ps@<pagestyle></code>	120	<code>\thispdfpagelabel</code>	46
<code>\ps@f@nch@fancyproto</code>	120	<code>\thmt@allenvs</code>	120
<code>\put</code>	27	<code>\thmt@contentsline</code>	120
<code>\raggedcolumns</code>	24	<code>\thmt@contentslineShow</code>	120
<code>\ReadonlyShipoutCounter</code>	46	<code>\thmt@mklistcmd</code>	120
<code>\ref</code>	44	<code>\thmt@numberline</code>	120
<code>\refstepcounter@noarg</code>	121	<code>\thmt@shortoptarg</code>	120
<code>\refstepcounter@optarg</code>	121	<code>\thmt@thmname</code>	120
<code>\relax</code>	4, 43, 88, 98	<code>\thmt@formatoptarg</code>	120
<code>\renewcommand</code>	46	<code>\thmtlo@newentry</code>	120
<code>\reversemarginpar</code>	82	<code>\tikz@finish</code>	121
<code>\rightmark</code>	59	<code>\tikz@path@do@at@end</code>	121
<code>\RL@column@boxes</code>	120	<code>\title@class@<class></code>	59
<code>\RLmulticolcolumns</code>	26	<code>\title@classhook@<class>afterdef</code>	59
<code>\section</code>	23, 60, 61, 68, 69, 121	<code>\title@classhook@<class>begin</code>	59
<code>\sectionmark</code>	34	<code>\title@classhook@<class>end</code>	59
<code>\set@keptmarks</code>	122	<code>\title@classinitial@<class></code>	59
<code>\setcolumnwidth</code>	81	<code>\title@classkeys@<class></code>	59
<code>\setgraphicspath</code>	4, 43, 70, 84	<code>\title@ifstar</code>	59
<code>\shadecontent</code>	70	<code>\tmcbl@beforepenalty</code>	37
<code>\shadecontentbox</code>	70	<code>\tmcbl@beforeskip</code>	37
<code>\shadetext</code>	70	<code>\tmcbl@code</code>	38
<code>\shadetextbox</code>	70	<code>\tmcbl@code@</code>	39
<code>\skip</code>	13	<code>\tmcbl@format</code>	37
<code>\string</code>	75	<code>\tmcbl@hang</code>	37
<code>\subparagraph</code>	69, 121	<code>\tmcbl@hyper</code>	38
<code>\subsection</code>	69, 121	<code>\tmcbl@hyper@</code>	39
<code>\subsubsection</code>	69, 121	<code>\tmcbl@hyperlink</code>	39
<code>\tableofcontents</code>	55, 60	<code>\tmcbl@hyperlinkbox</code>	39

<code>\tmcbl@hyperrange</code>	38	<code>I</code>	79
<code>\tmcbl@ignore</code>	38	<code>Iformat</code>	78
<code>\tmcbl@indent</code>	37	<code>label-prefix</code>	79
<code>\tmcbl@leader</code>	38	<code>label-suffix</code>	79
<code>\tmcbl@leader@</code>	39	<code>N</code>	79
<code>\tmcbl@leadercontent</code>	38	<code>Nformat</code>	78
<code>\tmcbl@leaderoptions</code>	38	<code>Nleft</code>	79
<code>\tmcbl@leadersep</code>	38	<code>Nright</code>	79
<code>\tmcbl@leftskip</code>	37	<code>O</code>	79
<code>\tmcbl@linewidth</code>	38	<code>Oformat</code>	78
<code>\tmcbl@name</code>	38	<code>T</code>	79
<code>\tmcbl@name@</code>	39	<code>Tformat</code>	78
<code>\tmcbl@nameformat</code>	37	<code>Tleft</code>	79
<code>\tmcbl@namewidth</code>	38	<code>Tright</code>	79
<code>\tmcbl@page</code>	38	<code>texnote</code>	76
<code>\tmcbl@page@</code>	39	text 的命令:	
<code>\tmcbl@pageformat</code>	38	<code>\text_expand:n</code>	6, 41
<code>\tmcbl@pagewidth</code>	38	<code>\text_mdfive_hash:n</code>	41, 119
<code>\tmcbl@right</code>	37	<code>\text_purify:n</code>	6
<code>\tmcbl@title</code>	38	<code>\thetitlecount</code>	31, 31, 34
<code>\tmcbl@title@</code>	39	<code>\thetotal<counter>s</code>	84
<code>\tmcbl@titleformat</code>	38	<code>\thetotal<counter>x</code>	84
<code>\tmcbl@titlewidth</code>	38	title/... 的选项:	
<code>\tmcbltheanchor</code>	39	<code>afterindent</code>	33
<code>\tmcbltheclasse</code>	39	<code>aftername</code>	32
<code>\tmcbltheinfo</code>	39	<code>aftername+</code>	32
<code>\tmcblthelevel</code>	39	<code>afterskip</code>	33
<code>\tmcblthename</code>	39	<code>aftertitle</code>	32
<code>\tmcblthepage</code>	39	<code>aftertitle+</code>	32
<code>\tmcblthetitle</code>	39	<code>beforerecord</code>	33
<code>\tmcblthetype</code>	39	<code>beforerecord<</code>	33
<code>\topskip</code>	12	<code>beforerecord></code>	33
<code>\unexpanded</code>	49, 52	<code>beforeskip</code>	33
<code>\unhbox</code>	54, 55	<code>bookmark</code>	34
<code>\UseName</code>	56	<code>bookmark*</code>	34
<code>\usethispagestyle</code>	15	<code>bookmark-extra</code>	34
<code>\vbox</code>	85–87	<code>break</code>	33
<code>\verb</code>	9	<code>break+</code>	33
<code>\voffset</code>	14	<code>ensureskip</code>	33
<code>\vspace</code>	20	<code>fixskip</code>	33
<code>\vsplit</code>	85–87, 97	<code>float-barrier</code>	33
<code>\vtop</code>	85	<code>format</code>	32
<code>\xleaders</code>	21	<code>format+</code>	32
<code>\xspace</code>	84	<code>hang</code>	32
texbnf 的选项:		<code>indent</code>	32
<code>clear-all-format</code>	78	<code>label-format</code>	31, 32
<code>format</code>	78	<code>leftskip</code>	33
<code>hyper</code>	77	<code>level</code>	31, 31
<code>hyper-color</code>	77	<code>mark</code>	34

[illegible]

List of Hackings

本章列出 CuS_TE_X 宏集中使用的内核内部命令、其它宏包的内部命令以及所有 hacking。

为 L^AT_EX3 的 l3text 增加了: \text_mdive_hash:n;

为 L^AT_EX3 的 l3token 增加了: \token_if_cs_word:NTF, \token_if_control_word:NTF;

cus.module.ltx.tex

使用了 L^AT_EX2_ε 内核的内部命令 (或环境): \@dischph;

使用了 L^AT_EX2_ε 内核的内部命令 (或环境): \input@path;

使用了 graphicx 的内部命令 (或环境): \Ginput@path, \Ginclude@graphics;

为 graphicx 增加了: \IfGraphicsExists, \InputIfGraphicsExists;

cus.module.util.tex

使用了 L^AT_EX3 的 l3seq 的内部命令 (或环境): __seq_push_item_def:n, __seq_pop_item_def:, \s__seq, __seq_item:n;

使用了 L^AT_EX2_ε 内核的内部命令 (或环境): \c@page;

使用了 L^AT_EX2_ε 内核的内部命令 (或环境): \@onlypreamble, \@notprerr;

使用了 L^AT_EX2_ε 内核的内部命令 (或环境): \@twoclasseserror;

为 L^AT_EX3 的 l3keys 增加了: .switch_set:N, .clist_put_right:N, .clist_put_left:N, .toks_put_right:N, .toks_put_left:N, .obey_psrerule:nn, .gbey_psrerule:nn;

使用了 hyperref 的内部命令 (或环境): \@filecolor, \@linkcolor, \@citecolor, \@urlcolor;

使用了 L^AT_EX2_ε 内核的内部命令 (或环境): \@auxout, \@currentlabel;

patch 了 bookmark 的命令 (或环境): \BKM@hook;

修改了 bookmark 的命令 (或环境): \bookmark@text;

使用了 hyperref 的内部命令 (或环境): \@currentHref, \ifHy@pdfstring;

使用了 hyperref 的内部命令 (或环境): \hyper@anchor, \hyper@anchorstart, \hyper@anchorend, \hyper@link, \hyper@linkstart, \hyper@linkend, \hyper@linkfile, \hyper@linkurl;

使用了 pdfmanagement-testphase 的内部命令 (或环境): \hyper@linklaunch, \hyper@linknamed;

使用了 hyperref 的内部命令 (或环境): __hyp_target_create_bool, __hyp_target_manual:nn, __hyp_target_manual:nn, __hyp_target_counter_anon:n;

使用了 hyperref 的内部命令 (或环境): __hyp/target/setname;

patch 了 hyperref 的命令 (或环境): \HyPL@EveryPage;

修改了 hyperref 的命令 (或环境): \HyPL@page, \HyPL@thisLabel;

使用了 hyperref 的内部命令 (或环境): \Hy@SaveSpaceFactor, \Hy@RestoreSpaceFactor;

cus.module.algo.tex

使用了 L^AT_EX3 的 l3tl 的内部命令 (或环境): __tl_map_function:Nnnnnnnnn, __tl_map_tokens:nnnnnnnn \s__tl_stop;

cus.module.layout.tex

使用了 geometry 的内部命令 (或环境): \Gm@warning;

使用了 geometry 的内部命令 (或环境): \Gm@save;

使用了 geometry 的内部命令 (或环境): \Gm@restore;

为 geometry 增加了: \Gm@..paper;

使用了 geometry 的内部命令 (或环境): \Gm@setsize;

使用了 geometry 的内部命令 (或环境): \Gm@changelayout;

使用了 fancyhdr 的内部命令 (或环境): \f@nch@0@olh, \f@nch@0@orh, \f@nch@0@elh, \f@nch@0@erh;

使用了 fancyhdr 的内部命令 (或环境): \f@nch@0@olf, \f@nch@0@orf, \f@nch@0@elf, \f@nch@0@erf;

修改了 L^AT_EX2_ε 内核的命令 (或环境): \pagestyle, \thispagestyle

patch 了 L^AT_EX2_ε 内核的命令 (或环境): \@outputpage

使用了 L^AT_EX 2_ε 内核的内部命令 (或环境): `\ps@{pagestyle}`, `\is@specialpage`, `\@specialstyle`
 使用了 fancyhdr 的内部命令 (或环境): `\f@nch@pagestyle`;
 使用了 fancyhdr 的内部命令 (或环境): `\f@nch@headinit`, `\f@nch@footinit`;
 使用了 fancyhdr 的内部命令 (或环境): `\ps@f@nch@fancyproto`, `\f@nch@def`, `\f@nch@ps@{pagestyle}-is-fancyhdr`;
 使用了 L^AT_EX 2_ε 内核的内部命令 (或环境): `\if@fcolmade`;
 使用了 fancyhdr 的内部命令 (或环境): `\iff@nch@footnote`;

cus.module.box.tex

使用了 L^AT_EX 2_ε 内核的内部命令 (或环境): `\if@in@minipage@env`;
 使用了 L^AT_EX 2_ε 内核的内部命令 (或环境): `\@parboxrestore`, `\@mpfn`, `\@minipagerestore`, `\@setminipage`;
 使用了 varwidth 的内部命令 (或环境): `\@vwid@setup`, `\@vwid@minipagerestore`;
 使用了 L^AT_EX 2_ε 内核的内部命令 (或环境): `\@iiiparbox`;
 使用了 L^AT_EX 2_ε 内核的内部命令 (或环境): `\if@nobreak`;
 使用了 L^AT_EX 2_ε 内核的内部命令 (或环境): `\@dischyph`, `\if@noskipsec`;
 使用了 L^AT_EX 2_ε 内核的内部命令 (或环境): `\@acci`, `\@accii`, `\@acciii`;
 使用了 L^AT_EX 2_ε 内核的内部命令 (或环境): `\@totalleftmargin`;
 使用了 multicol 的内部命令 (或环境): `\mult@rightbox`, `\mult@nat@firstbox`, `\mult@rightbox`;
 patch 了 multicol 的命令 (或环境): `\page@sofar`, `\prepare@multicols`, `\multi@column@out`;
 hack 了 multicol 的命令 (或环境): `\LR@column@boxes`, `\RL@column@boxes`, `\mc@align@columns`;
 使用了 multicol 的内部命令 (或环境): `\c@minrows`, `\c@unbalance`, `\c@columnbadness`, `\c@finalcolumnbadness`;
 使用了 rotating 的内部命令 (或环境): `\Grot@setangle`;
 使用了 L^AT_EX 2_ε 内核的内部命令 (或环境): `\color@hbox`;
 使用了 rotating 的内部命令 (或环境): `\@rotfloat`, `\end@rotfloat`, `\@rotdblfloat`, `\end@rotdblfloat`;

cus.module.bgfg.tex

使用了 L^AT_EX 2_ε 内核的内部命令 (或环境): `\c@page`;
 使用了 geometry 的内部命令 (或环境): `\Gm@layoutwidth`, `\Gm@layoutheight`;

cus.module.index.tex

修改了 L^AT_EX 2_ε 内核的命令 (或环境): `\@indexfile`;
 为 L^AT_EX 2_ε 内核增加了: `\@indexfile..`;
 使用了 L^AT_EX 2_ε 内核的内部命令 (或环境): `\@idxitem`;
 修改了 L^AT_EX 2_ε 内核的命令 (或环境): `\index`, `\makeindex`, `\printindex`;
 使用了 L^AT_EX 2_ε 内核的内部命令 (或环境): `\@bsphack`, `\@sanitize`, `\@wrindex`;
 使用了 L^AT_EX 2_ε 内核的内部命令 (或环境): `\@input@`;

cus.module.struct.tex

使用了 L^AT_EX 2_ε 内核的内部命令 (或环境): `\@input`, `\@writefile`, `\@auxout`;
 使用了 L^AT_EX 2_ε 内核的内部命令 (或环境): `\@auxout`, `\tf@toc`;
 使用了 L^AT_EX 2_ε 内核的内部命令 (或环境): `\@kernel@after@enddocument@afterlastpage`;
 使用了 hyperref 的内部命令 (或环境): `\Hy@tocdestname`, `\Hy@linktoc`, `\hyper@link`;
 patch 了 float 的命令 (或环境): `\newfloat`, `\floatplacement`;
 patch 了 newfloat 的命令 (或环境): `\@DeclareFloatingEnvironment`, `\newfloat@setoptions`;
 patch 了 floatrow 的命令 (或环境): `\DeclareNewFloatType`, `\FB@capttype`;
 patch 了 tcolorbox 的命令 (或环境): `\tcb@proc@options@init`, `\kv tcb@new@listof`, `\kv tcb@new@listtype`;
 patch 了 amsthm 的命令 (或环境): `\@xnthm`;
 修改了 thmtools 的命令 (或环境): `\thmt@mklistcmd`, `\listoftheorems`;
 使用了 thmtools 的内部命令 (或环境): `\thmtlo@newentry`, `\ifthmt@isstarred`, `\ll@<thmt envname>`, `\thmt@thmname`, `\thmt@shortoptarg`, `\thmt@formatoptarg`, `\thmt@contentslineShow`;
 使用了 thmtools 的内部命令 (或环境): `\thmt@numberline`, `\ifthmt@listswap`, `\thmt@allenvs`, `\thmtlo@newentry`;
 修改了 thmtools 的命令 (或环境): `\thmt@contentsline`;

修改了 `hyperref` 的命令 (或环境): `\addtocontents, \contentsline`;
 修改了 `ctexheading` 的命令 (或环境): `\CTEX@addloflotskip`;
 使用了 L^AT_EX_{2_ε} 内核的内部命令 (或环境): `\f@size`;
 使用了 L^AT_EX_{2_ε} 内核的内部命令 (或环境): `\c@tocdepth, \if@restonecol`;
 修改了 L^AT_EX_{2_ε} 内核的命令 (或环境): `\@starttoc`;
 使用了 L^AT_EX_{2_ε} 内核的内部命令 (或环境): `\e@alloc@chardef, \allocationnumber`;
 使用了 L^AT_EX_{2_ε} 内核的内部命令 (或环境): `\toclevel@...`;
 patch 了 `hyperref` 的命令 (或环境): `\H@refstepcounter`;
 patch 了 `cleveref` 的命令 (或环境): `\refstepcounter@noarg, \refstepcounter@optarg`;
 使用了 `nameref` 的内部命令 (或环境): `\NR@gettitle`;
 为 `hyperref` 增加了: `\hyper@nopatch@sectioning`;
 为 `nameref` 增加了: `\NR@nopatch@sectioning`;
 使用了 L^AT_EX_{2_ε} 内核的内部命令 (或环境): `\c@secnumdepth`;
 修改了 L^AT_EX_{2_ε} 内核的命令 (或环境): `\part, \chapter, \section, \subsection, \subsubsection, \paragraph, \subparagraph`;
 使用了 L^AT_EX_{2_ε} 内核的内部命令 (或环境): `\@topnewpage`;
 使用了 L^AT_EX_{2_ε} 内核的内部命令 (或环境): `\@maketophead, \@makestophead, \@afterheading`;
 使用了 `placeins` 的内部命令 (或环境): `\FloatBarrier`;
 使用了 L^AT_EX_{2_ε} 内核的内部命令 (或环境): `\@svsec, \@svsechd`;
 使用了 L^AT_EX_{2_ε} 内核的内部命令 (或环境): `\@secpenalty`;

cus.library.box.tex

为 L^AT_EX_{2_ε} 内核增加了: `\@tempdimd`;
 为 `longfbox` 增加了: `/longfbox/math, /longfbox/highlight math`;
 使用了 `paracol` 的内部命令 (或环境): `\ifpcol@swapcolumn, \ifpcol@swapmarginpar, \ifpcol@bg@swap`;
 使用了 L^AT_EX_{2_ε} 内核的内部命令 (或环境): `\cl@ckpt`;
 使用了 `paracol` 的内部命令 (或环境): `\pcol@gcounters, \pcol@globalcounter`;
 使用了 `paracol` 的内部命令 (或环境): `\pcol@colwidthspecleft, \pcol@colwidthspecright, \pcol@columnratioleft, \pcol@columnratioreight`;
 使用了 `paracol` 的内部命令 (或环境): `\pcol@mpthresholdl, \pcol@mpthresholdr`;

cus.library.math.tex

使用了 L^AT_EX_{2_ε} 内核的内部命令 (或环境): `\frozen@everymath, \frozen@everydisplay`;
 使用了 `amsmath` 的内部命令 (或环境): `\bBigg@`;

cus.library.counter.tex

使用了 L^AT_EX_{2_ε} 内核的内部命令 (或环境): `\cl@ckpt, \@elt`;
 修改了 L^AT_EX_{2_ε} 内核的命令 (或环境): `\@stpelt`;

cus.library.ref.tex

使用了 L^AT_EX_{2_ε} 内核的内部命令 (或环境): `\c@page, \ifG@refundefined`;

cus.library.pgf.tex

使用了 `pgf` 的内部命令 (或环境): `\pgf@declareimage`;
 使用了 `tikz` 的内部命令 (或环境): `\tikz@finish, \tikz@path@do@at@end`;
 使用了 `fancyhdr` 的内部命令 (或环境): `\f@nch@O@olh, \f@nch@O@orh, \f@nch@O@elh, \f@nch@O@erh`;
 使用了 `fancyhdr` 的内部命令 (或环境): `\f@nch@O@olf, \f@nch@O@orf, \f@nch@O@elf, \f@nch@O@erf`;

cus.library.tcb.tex

使用了 `tcolorbox` 的内部命令 (或环境): `\kv tcb@boxsep`;

使用了 `tcolorbox` 的内部命令 (或环境): `\kv tcb@top@rule@stand`, `\kv tcb@bottom@rule@stand`, `\kv tcb@top`, `\kv tcb@bottom`;

使用了 `tcolorbox` 的内部命令 (或环境): `\kv tcb@left@rule@stand`, `\kv tcb@right@rule@stand`, `\kv tcb@leftupper`, `\kv tcb@rightupper`;

cus.library.pdf.tex

使用了 `graphicx` 的内部命令 (或环境): `\Gin@ii`;

使用了 `newpax` 的内部命令 (或环境): `\NEWPAX@cmd@<type>`, `\NEWPAX@skip`, `\NEWPAX@includegraphics`, `\NEWPAX@AddAnnots`, `\NEWPAX@file`, `\NEWPAX@Gin@page`;

It3ekeys、It3ekeyscmd 和 It3ekeysext

使用了 $\text{\LaTeX} 2_{\epsilon}$ 内核的内部命令 (或环境): `__cmd_peek_nonspace:NTF`, `__cmd_peek_nonspace_remove:NTF`;

使用了 $\text{\LaTeX} 2_{\epsilon}$ 内核的内部命令 (或环境): `__cmd_token_if_cs:NTF`;

使用了 $\text{\LaTeX} 3$ 的 `l3keys` 的内部命令 (或环境): `\c__keys_type_root_str`, `\c__keys_code_root_str`, `\c__keys_props_root_str`, `\c__keys_inherit_root_str`;

使用了 $\text{\LaTeX} 2_{\epsilon}$ 内核的内部命令 (或环境): `\e@alloc@top`;

使用了 $\text{\LaTeX} 3$ 的 `l3keys` 的内部命令 (或环境): `\l__keys_module_str`, `\l_keys_path_tl`, `\l__keys_inherit_str`;

修改了 $\text{\LaTeX} 2_{\epsilon}$ 内核的命令 (或环境): `\@showcommandlisthook`, `\@declarecommandcopylisthook`, `\g_hook_patch_action_list_tl`;

使用了 $\text{\LaTeX} 2_{\epsilon}$ 内核的内部命令 (或环境): `__hook_patch_expand_redefine:NNnn`;

It3ekeys-elkernel

使用了 $\text{\LaTeX} 2_{\epsilon}$ 内核的内部命令 (或环境): `__kernel_cmd_if_xparse:NTF`;

使用了 $\text{\LaTeX} 3$ 内核的内部命令 (或环境): `__kernel_str_to_other_fase:n`;

使用了 $\text{\LaTeX} 3$ 内核的内部命令 (或环境): `__kernel_chk_if_free_cs:N`;

使用了 $\text{\LaTeX} 3$ 内核的内部命令 (或环境): `__kernel_quark_new_test:N`, `__kernel_quark_new_conditional:Nn`;

使用了 $\text{\LaTeX} 3$ 内核的内部命令 (或环境): `__kernel_backend_literal_pdf:e`;

使用了 $\text{\LaTeX} 3$ 内核的内部命令 (或环境): `__kernel_file_name_sanitize:n`;

It3ekeys-collectn

使用了 $\text{\LaTeX} 2_{\epsilon}$ 内核的内部命令 (或环境): `\e@alloc@chardef`;

updatemarks

使用了 $\text{\LaTeX} 2_{\epsilon}$ 内核的内部命令 (或环境): `\g__mark_classes_seq`;

使用了 $\text{\LaTeX} 2_{\epsilon}$ 内核的内部命令 (或环境): `__mark_update_structure_alias:nn`;

使用了 $\text{\LaTeX} 2_{\epsilon}$ 内核的内部命令 (或环境): `\g__mark_..._tl`, `\c__mark_class..._mark`;

patch 了 $\text{\LaTeX} 2_{\epsilon}$ 内核的命令 (或环境): `\endminipage`;

patch 了 `tcolorbox` 的命令 (或环境): `\tcbox@inner@box`, `\end tcb@lrbox`, `\end tcb@savebox`, `\tcbox@drawing@env@end`, `\tcbox@vsplit@upper`, `\tcbox@vsplit@lower`, `\tcbox@split@start`, `\tcbox@split@USL`, `\tcbox@split@SL@displayed`, `\tcbox@split@L`;

patch 了 `multicol` 的命令 (或环境): `\set@keptmarks`, `\endmulticols`, `\multi@column@out`, `\balance@columns@out`, `\balance@columns`, `\prep@keptmarks`;

patch 了 `adjmulticol` 的命令 (或环境): `\adjmc@process@ne@column`;