CusTeX 宏集手册

Longaster

2023 年 1 月 20 日　v0.0.2

总目录

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 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$_\varepsilon$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 ⟨*key path*⟩，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 来设置这些键值，需要指定 ⟨*key path*⟩。

\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。

§ 1 util 模块

util 模块。

\Replicate ⋆

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

重复 ⟨*code*⟩ ⟨*num expr*⟩ 次。

\MapClist ☆
\MapList ☆

```
\MapClist {⟨comma list⟩} {⟨tokens⟩}
\MapList  {⟨list⟩} {⟨tokens⟩}
```

\MapClist 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 使用 ⟨*tokens*⟩ 迭代记号列表 ⟨*list*⟩，它将 ⟨*tokens*⟩ 置于列表项之前。

\IterateClist
\IterateList

```
\IterateClist {⟨comma list⟩} {⟨inline code⟩}
\IterateList  {⟨list⟩} {⟨inline code⟩}
```

\IterateCist 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 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。

例 1

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

$a_1 a_2 a_3 a_n$ $a_1 + a_2 + a_3 + a_n$

\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。

例 2

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

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

例 3

```
$ \IterateThread[2]{1,2,3,n}{n,n-1,n-2,1}[+][+\cdots+]{a_{#2}\cdot b^{#3}} $
%= $ 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 ☆
\ucchars ☆

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

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

例 4

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

如：util 模块。

\zkern

相当于 \kern\z@。

\Verbatimize

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

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
\ifAbsPageOdd

```
\ifPageOdd {⟨true⟩} {⟨false⟩}
```

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/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 5.3 节。

§ 2 页面布局，layout 模块

layout 提供页面布局的相关接口。

\setuplayout

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

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。

2.2.1 页面尺寸

layout/papername
layout/paper

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

设置纸张大小。⟨*papername*⟩ 为预定义的纸张名，大小写无关。

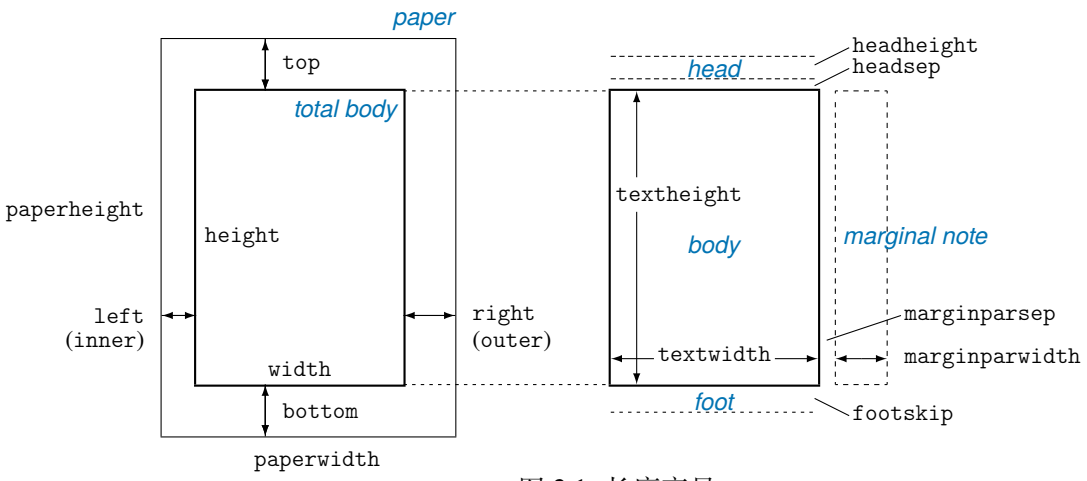


图 2.1: 长度变量

```
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。

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

```
paperorientation|orientation = ⟨landscape|portrait⟩
landscape                                          不可设置值
portrait                                           不可设置值
direction = ⟨bigwidth|bigheight|normal|inverse⟩
```

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

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/layouthoffset
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。

2.2.2 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⟩}                                              初始值：0.7
vscale = {⟨正实数⟩}                                              初始值：0.7
scale  = {⟨hscale⟩,⟨vscale⟩} 或 {⟨正实数⟩}
```

设置 *total part* 部分的宽高与纸张宽高的比率。

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

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

设置 *total part* 部分的宽高。

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

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

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

layout/lines

```
lines = {⟨行数⟩}
```

根据 ⟨行数⟩ 设置 textheight。⟨行数⟩ 一般为正整数。

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

```
includehead = ⟨true|false⟩                                       初始值：false
includefoot = ⟨true|false⟩                                       初始值：false
includeheadfoot|includehf = ⟨true|false⟩
```

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

layout/includemarginpar
layout/includemp

```
includemarginpar|includemp = ⟨true|false⟩                        初始值：false
```

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

layout/includeall

```
includeall = ⟨true|false⟩                                        初始值：false
```

设置 includeheadfoot 及 includemarginpar。

layout/ignorehead
layout/ignorefoot
layout/ignoreheadfoot
layout/ignorehf

```
ignorehead = ⟨true|false⟩                                        初始值：false
ignorefoot = ⟨true|false⟩                                        初始值：false
ignoreheadfoot|ignorehf = ⟨true|false⟩
```

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。

layout/ignoremarginpar
layout/ignoremp

```
ignoremarginpar|ignoremp = ⟨true|false⟩                          初始值：false
```

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。

`ignoreall = ⟨true|false⟩` 初始值：`false`

layout/ignoreall

设置 ignoreheadfoot 及 ignoremarginpar。

`heightrounded = ⟨true|false⟩` 初始值：`false`

layout/heightrounded

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。

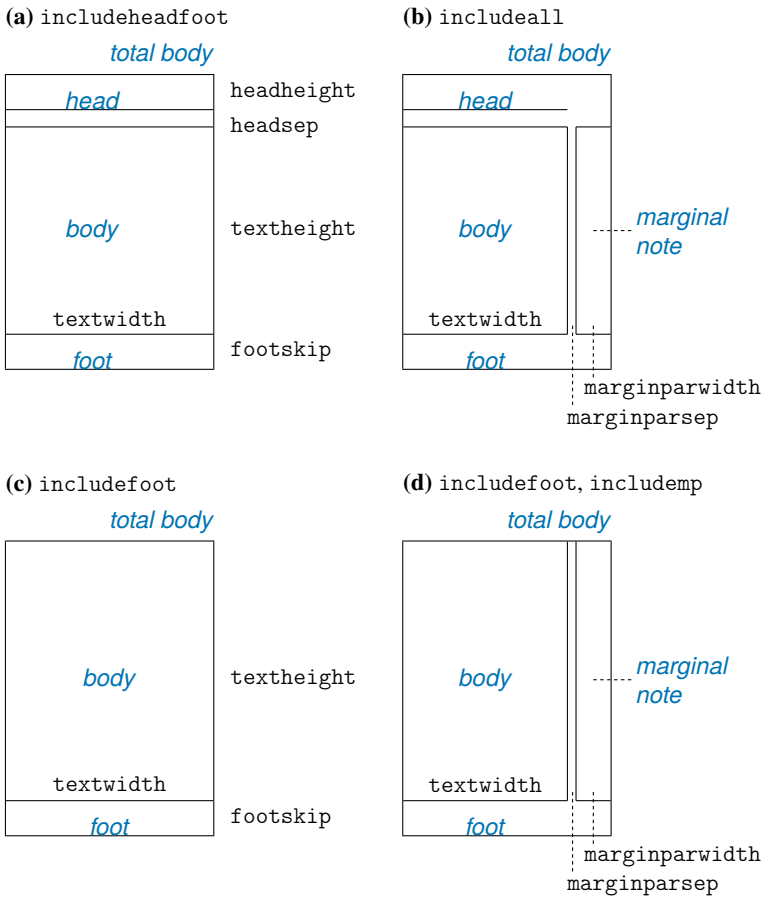


图 2.2
不同模式下的 *total body*。(a) includeheadfoot，(b) includeall，(c) includefoot 及 (d) includefoot，includemp。如果 reversemarginpar 设置为真，则交换 *marginal note* 与 *body* 的位置。如果设置了 twoside，则依据奇偶页交换 *marginal note*。

`hdivide = {⟨left margin⟩,⟨width⟩,⟨right margin⟩}`
`vdivide = {⟨top margin⟩,⟨height⟩,⟨bottom margin⟩}`
`divide = {⟨length$_1$⟩,⟨length$_2$⟩,⟨length$_3$⟩}`

layout/hdivide
layout/vdivide
layout/divide

设置两个值，将另一个留空或 *。

2.2.3 边距

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

```
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。

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

```
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。

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

```
hmarginratio|horizontalmarginratio = {⟨inner ration⟩}:{⟨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。

layout/hcentering
layout/vcentering
layout/centering

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

设置 hmarginratio、vmarginratio 为 1:1。

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

```
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。

layout/bindingoffset

```
bindingoffset = {⟨长度⟩}
```

从内侧移除 ⟨长度⟩。

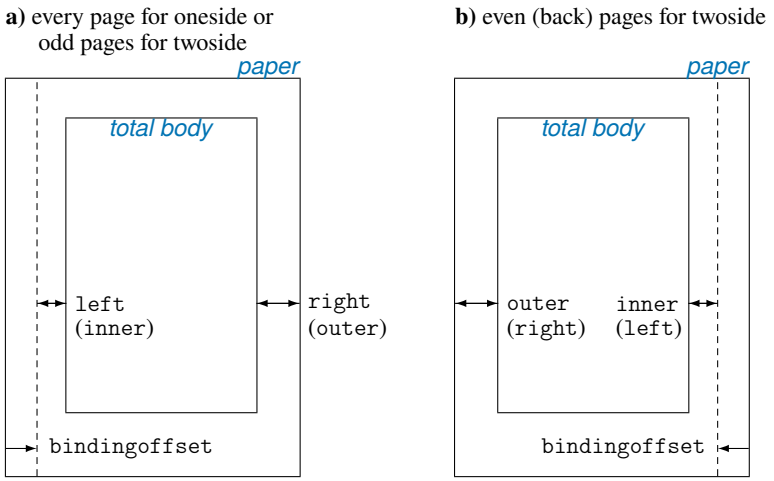


图 2.3

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).

2.2.4 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⟩}`

layout/footnotesep

设置 \skip\footins，即正文底部与脚注顶部的距离。

`marginparwidth|marginpar = {⟨长度⟩}`
`marginparsep = {⟨长度⟩}`
`nomarginpar|nomp` 不可设置值

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

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

`columnsep = {⟨长度⟩}`
`twocolumn` 不可设置值
`onecolumn` 不可设置值

layout/columnsep
layout/twocolumn
layout/onecolumn

设置 \columnsep，即两栏之间的距离。

`hoffset = {⟨长度⟩}`
`voffset = {⟨长度⟩}`
`offset = {⟨hoffset⟩,⟨voffset⟩}` 或 `{⟨长度⟩}`

layout/hoffset
layout/voffset
layout/offset

设置 \hoffset、\voffset。

2.2.5 页眉页脚

`head|headheight = {⟨长度⟩}`
`headsep = {⟨长度⟩}`
`nohead`

layout/headheight
layout/head
layout/headsep

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。

`footskip|foot = {⟨弹性长度⟩}`

layout/footskip
layout/foot
layout/nofoot

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。

`noheadfoot|nohf` 不可设置值

layout/noheadfoot
layout/nohf

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。

2.2.6 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。

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

```
showframe = ⟨true|false⟩                          初始值：false
showcrop  = ⟨true|false⟩                          初始值：false
showmarking|marking = ⟨true|false⟩                初始值：false
```

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。

layout/preset
layout/name

```
preset|name = {⟨preset name⟩}
```

使用预设值 ⟨*preset name*⟩。

2.2.7 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 2.2.5 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

```
\usepagestyle {⟨pagestyle⟩}
```

使用页眉页脚的样式 ⟨*pagestyle*⟩。

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

\setpagestyle

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

设置样式 {⟨*pagestyle*⟩}，或基于样式 ⟨*pagestyle*$_2$⟩ 设置 ⟨*pagestyle*$_1$⟩。

带 * 的，仅设置而不使用。不带 * 的，还会立刻使用该样式。

\sethead
\setfoot
\setheadfoot
\setlefthead
\setcenterhead
\setrighthead
\setleftfoot
\setcenterfoot
\setrightfoot

```
\sethead          {⟨code⟩}
\sethead [⟨位置⟩] {⟨code⟩}
\setcenterhead           {⟨奇偶页⟩}
\setcenterhead [⟨偶数页⟩] {⟨奇数页⟩}
```

设置页眉页脚的内容。

⟨位置⟩ 为 O、E、L、C、R、H、F 此三类的组合。这七个个值分别代表奇偶、左中右、页眉页脚。不区分大小写。

如某一类未给出，则视为该类的全部值都给出。但 \sethead、\setfoot 分别为 H、F。

例如，在 \sethead 中，L 代表 OLF,ELF。

它们可以直接用在导言区和正文中，将修改本页及其后页面的页眉页脚。但最好用于 \setpagestyle 命令中，统一设置页眉页脚。

\setheadrulewidth
\setfootrulewidth

```
\setheadrulewidth {⟨长度表达式⟩}
```

设置页眉、页脚横线的厚度。

（即宏 \headrulewidth、\footrulewidth 的值。）

\setheadruleskip {⟨skip expr⟩}

\setheadruleskip
\setfootruleskip

设置页眉、页脚横线与页眉、页脚文字的距离。

（即宏 \headruleskip、\footruleskip 的值。）

\setheadrule {⟨code⟩}

\setheadrule
\setfootrule

设置页眉、页脚的横线。页眉的横线的总高度最好为 0。

\setheadinit {⟨code⟩}

\setheadinit
\setfootinit
\setheadfootinit

在输出页眉页脚前要执行的 ⟨code⟩。

\fancycenter {⟨left⟩} {⟨center⟩} {⟨right⟩}
\fancycenter [⟨distance⟩] [⟨stretch⟩] {⟨left⟩} {⟨center⟩} {⟨right⟩}

\fancycenter

它创建一个盒子，使得 ⟨center⟩ 位于当前行（或盒子）的中心。可以用于正文中。

⟨center⟩ 的中心与 ⟨left⟩、⟨right⟩ 的中心的距离可能并不一致。

例 5

```
\fancycenter{L}{CCCC}{RRRRRRRRRRRRRRR}
```

L CCCC RRRRRRRRRRRRRRR

\iftopfloat {⟨true⟩} {⟨false⟩}

\iftopfloat ⋆
\ifbotfloat ⋆
\iffloatpage ⋆
\iffootnote ⋆

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。

§ 3 盒子和填充，box 模块

box 用于提供盒子构造、内容填充等内容。

2.3.1 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。

\begin{Framed} [⟨frame key-val⟩]
...
\end{Framed}

Framed

创建一个可跨页的盒子。若在另一个盒子内则不可跨页。

outer-sep = {⟨skip expr⟩} 初始值：8pt plus 8pt minus 6pt

frame/outer-sep

设置盒子与上下文的间距。

sep = {⟨长度表达式⟩} 初始值：3\fboxsep

frame/sep

设置变量 \cusframesep，即盒子外框与内容的间距。

rule-width = {⟨长度表达式⟩} 初始值：\fboxrule

frame/rule-width

设置变量 \cusframerule，即盒子外框的厚度。

frame/frame
frame/frame*
frame/first
frame/first*
frame/middle
frame/middle*
frame/last
frame/last*
frame/whole
frame/whole*

frame = {⟨*code*⟩}
frame* = {⟨*code width 1 parameter*⟩}

frame 设置盒子外框。

first，middle，last 设置分页盒子的前、中、后三部分的外框。

whole 设置未分页盒子的外框。

⟨*code*⟩ 其后可接一个参数，这个参数为分页后的盒子。⟨*code with 1 parameter*⟩ 显式给出变量 #1。

frame/init

init = {⟨*code*⟩}

盒子中执行的初始化代码。

frame/width
frame/ratio

width = {⟨长度表达式⟩} 初始值：\textwidth
ratio = {⟨数值表达式⟩} 初始值：1

ratio 设置盒子内容（含边框）占 width 的比率。

frame/align
frame/left
frame/center
frame/right
frame/inner
frame/outer

align = ⟨left|center|right|inner|outer⟩ 初始值：center

设置水平对齐方式。当 $0 < \langle ratio \rangle < 1$ 时才有效。

例 6

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

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。

frame/ignore-warnings

ignore-warnings 不可设置值

忽略部分警告。

2.3.2 Filler

“filler” 是用以填充空间的那部分内容。如 LaTeX 2ε 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
\dashfiller {⟨filler width⟩}
\dashfiller [⟨raise⟩][⟨sep width⟩][⟨rule width⟩]
\dashfiller [⟨raise⟩] {⟨filler width⟩}[⟨sep width⟩][⟨rule width⟩]

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。

例 7

```
\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
```

\filler

\filler [⟨filler key-val⟩]

使用给定内容填充。

filler/size
filler/size*

size = {⟨skip expr⟩}
size* = {⟨长度⟩}

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=\hspace*{⟨*skip expr*⟩}。

vspace 相当于设置 space=\vspace{⟨*skip expr*⟩}。

vspace* 相当于设置 space=\vspace*{⟨*skip expr*⟩}。

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。

例 8

```
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 右间隔约 5cm。

左 右拉开。

左 中 右拉开。

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                                   不可设置值
clear-box                                       不可设置值
not-content                                     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。

filler/dash
filler/sep
filler/rule
filler/raise
filler/full
filler/is-dash

```
dash|sep = {⟨dash length⟩}                      初始值：0pt
rule     = {⟨rule width⟩}                       初始值：0.4pt
raise    = {⟨raise height⟩}                     初始值：0pt
full     = ⟨true|false⟩                         初始值：false
is-dash                                         不可设置值
```

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。

filler/solid
filler/dashed
filler/dotted
filler/cdotted

```
solid                                           不可设置值
dashed  = {⟨长度⟩}
dotted  = {⟨间距⟩}
cdotted = {⟨间距⟩}
```

使用实线、虚线、句点或 \cdot 填充。

例 9

```
\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
```

type = ⟨align|center|spread⟩ 初始值：align
align 不可设置值
center 不可设置值
spread 不可设置值

filler/type
filler/align
filler/center
filler/spread

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 {⟨*dim expr*⟩}
\atleastfiller [⟨*filler key-val*⟩] {⟨*dim expr*⟩}

\atleastfiller

填充的长度至少为 ⟨*dim epxr*⟩，太短的将自动断行。

例 10

```
我能吞下玻璃而不伤身体，I can eat glass, it dosen't hurt me.%
\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 [⟨*filler key-val*⟩]

\breakablefiller

可自动断行的 filler。

例 11

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

\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。

例 12

```
\newcommand\filllines[4][]{{% filler key-val, before, lines, after
  #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}{
↪ 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。

2.3.3 多栏文字

CusTeX 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 5.4 节。

在 multicol 中，可以使用

```
\begin{multicols}{col}                                          例 13
  ...
\end{multicols}
```

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
\stopmulticolumns

```
\startmulticolumns [⟨multicolumns key-val⟩]
  ⟨content⟩
\stopmulticolumns
```

将 ⟨*content*⟩ 以多栏排版。

multicolumns/columns
multicolumns/cols

```
columns|cols = {⟨整数表达式⟩}                                   初始值：2
```

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。

`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
`last-minimal = {⟨post length⟩}` 初始值：20pt

multicolumns/first-minimal
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 \section 等。它与其后的多栏文字保持在同一页。

`rule-width = {⟨length⟩}` 初始值：0pt
`rule-color = {⟨color⟩}`
`rule-color = [⟨color mode⟩] {⟨color⟩}`

multicolumns/rule-width
multicolumns/rule-color

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

`flush |aligned` 不可设置值
`ragged|not-aligned` 不可设置值

multicolumns/flush
multicolumns/aligned
multicolumns/ragged
multicolumns/not-aligned

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。

`balanced = ⟨true|false⟩` 初始值：true
`not-balanced` 不可设置值

multicolumns/balanced
multicolumns/not-balanced

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。

`columns*|cols* = {⟨栏数⟩}`

multicolumns/columns*
multicolumns/cols*

它在设置栏数的同时还设置 not-balanced。

注意 columns 并未决定使用 balanced 还是 not-balanced。

`swap-column = ⟨true|false⟩` 初始值：false
`enable-swap-column` 不可设置值
`disable-swap-column` 不可设置值

multicolumns/swap-column
multicolumns/enable-swap-column
multicolumns/disable-swap-column

控制是否在使用了 twoside 文档类选项时，偶数页逆序输出各栏。

enable-swap-column 用于启用此功能，disable-swap-column 用于禁用此功能。

multicolumns/framed
multicolumns/framed-option
multicolumns/framed-option+

framed = ⟨fbox|...⟩
framed-option = {⟨*options*⟩}

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-option 为可能的选项。

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 box 库提供 lfbox 值等，见第 5.4.2 小节。

使用 framed 选项可能存在分页失败的情况。

multicolumns/adj
multicolumns/adj-inner
multicolumns/adj-outer

adj = ⟨**true**|false⟩ 初始值：false
adj-inner = {⟨内侧长度⟩}
adj-outer = {⟨外侧长度⟩}

多栏文字还可以通过调整边距来调整总的文字宽度。adj 用于启用此功能。

⟨内侧长度⟩ 和 ⟨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。

multicolumns/addto-baselineskip

addto-baselineskip = {⟨*length*⟩}

设置 \multicolbaselineskip。

multicolumns/tolerance
multicolumns/pretolerance

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

设置 \multicoltolerance、\multicolpretolerance。

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

collectmore = {⟨*int expr*⟩}

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

multicolumns/top-fuzz
multicolumns/bottom-fuzz

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

设置 \multicolovershoot、\multicolundershoot。

multicolumns/v-fuzz
multicolumns/h-fuzz

v-fuzz = {⟨*length*⟩}

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

multicolumns/overflow

overflow = {⟨*dim expr*⟩} 初始值：12pt

设置 \maxbalancingoverflow。

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

left-to-right|LR 不可设置值
right-to-left|RL 不可设置值

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

2.3.4 旋转的盒子

CusTeX 封装了 rotating 宏包，提供了旋转的盒子。

```
\startrotate [⟨rotate key-val⟩]
  ⟨content⟩
\stoprotate

\Rotate [⟨rotate key-val⟩] {⟨content⟩}
```

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

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。

```
turn = {⟨number⟩}
rotate|nospaceturn = {⟨number⟩}
sideways                                    不可设置值
```

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

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`。

```
float  = {⟨float type⟩}
float* = {⟨float type⟩}
figure                                      不可设置值
figure*                                     不可设置值
```

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

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*⟩*。

§ 4 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.pdf`。

本手册前几页的水印使用如下代码实现：

例 14

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

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

\foreground
\background

```
\foreground   {⟨content⟩}
\foreground + {⟨content⟩}
\foreground (⟨位置⟩) {⟨content⟩}
\foreground [⟨hook label⟩] {⟨content⟩}
\foreground + (⟨位置⟩) [⟨hook label⟩] {⟨content⟩}
```

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；
- ⟨*hook label*⟩ 为 hook 的 label；此参数与 ⟨位置⟩ 的先后位置可交换；
  \foreground 默认为 ./fg，\background 默认为 ./bg；
  关于 hook label 的作用，请参考第 3.1 节或 lthooks.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。

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
\backgroundpicture

```
\foregroundpicture     {⟨图片⟩}
\foregroundpicture +   {⟨图片⟩}
\foregroundpicture   * {⟨图片⟩}
\foregroundpicture + * {⟨图片⟩}
\foregroundpicture + * (⟨位置⟩) [⟨hook label⟩] {⟨图片⟩}
```

将 ⟨图片⟩ 添加到前景或背景中。

+、⟨位置⟩、⟨*hook label*⟩ 的用法如前所述。

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 \background 放置背景图片。

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

例 15

如本页底部图片所示。

\removeforeground
\removebackground

```
\removeforeground
\removeforeground [⟨hook label⟩]
```

移除前景或背景。

§5 索引，index 模块

CusTeX 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 [⟨index keys⟩] {⟨index type⟩}
\setupindex [⟨index type list⟩] {⟨index keys⟩}

\newindextype
\setupindex

\newindextype 创建一个新的索引 ⟨index type⟩。\setupindex 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 [⟨index keys⟩]
\makeindex [⟨index keys⟩] [⟨index type⟩]

\makeindex

LaTeX 原有的接口。默认创建名称为空的索引。

⟨index keys⟩ 不同于 CuSTEX 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。

例 16

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

```
\index {⟨index item⟩}
\index [⟨index type⟩] {⟨index item⟩}
\printindex
\printindex [⟨index keys⟩]
```

\index 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；\printindex 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。

§6 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、ctexheading、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 CTEX 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

```
\definetitle {⟨title command⟩} {⟨title key-val⟩}
\definetitle {⟨title command⟩} [⟨title class⟩] {⟨title key-val⟩}
```

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 LaTeX book 类中，章节标题可分为三种，一种是以 \part 为代表的，CusTeX 将其命令为 page 类，一种是以 \chapter 为代表的，CusTeX 将其命名为 top 类，另一种则是以 \section 为代表，CusTeX 将其命名为 normal 类。[1] 另外还有 free、wrap 类，详细用法见第 4.1 节。

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。

\part
\chapter
\section
\subsection
\subsubsection
\paragraph
\subparagraph

```
\part   {⟨标题⟩}
\part * {⟨标题⟩}
\part   [⟨list entry⟩] {⟨标题⟩}
\part   [⟨title key-val⟩] {⟨标题⟩}
\part * [⟨title key-val⟩] {⟨标题⟩}
\part   [⟨title key-val⟩] [⟨list entry⟩] {⟨标题⟩}
```

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。

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

\setuptitle

```
\setuptitle {⟨title key-val⟩}
\setuptitle [⟨title list⟩] {⟨title key-val⟩}
```

设置标题的样式。⟨*title list*⟩ 为章节标题命令名称的列表，如：[chapter,section]，而非标题命令的列表。

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。

[1] 实际上，这些名称基本沿用了 titlesec 宏包的名称。

2.6.1 初始化设置

初始化设置选项可以在导言区修改任意次，但不可在正文设置。

```
number-from    = {⟨计数器⟩}
number-within  = {⟨计数器⟩}
number-without = {⟨计数器⟩}
```

title/.../number-from
title/.../number-within
title/.../number-without

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，这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。

```
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 *level*。在没有使用标题前，它展开为 −10001。

\CurrentTitleLevel ⋆

例如，现在它的值为 2。

2.6.2 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。

CuSTEX 预先定义了一些层级名称，如表 2.2 所示。

```
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 \setsecnumdepth 命令来设置。例如，对于 \section 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 level 键来修改层级）。因此 \section 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。

表 2.2: 层级名称

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

```
name = {⟨名字的前半部分⟩,⟨名字的后半部分⟩}
name = {⟨名字的前一部分⟩}
```

title/.../name

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。

```
number = {⟨数字输出命令⟩}
```

title/.../number

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 选项定义的同时将控制对章节计数器的交叉引用。在引用计数器时，记录在 LaTeX 辅助文件中的是 number 选项的定义。但是，number 选项不会影响计数器本身的输出。

2.6.3 格式

和 CTEX 宏集一样，CusTeX 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。

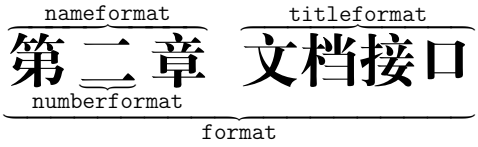


图 2.4: numberformat，nameformat，titleformat，format 几个选项的作用范围

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 \part）和 top （如 \chapter）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。关于页面样式的相关内容，见第 2.2.7 小节。

2.6.4 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。

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

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。

`fixskip = ⟨true|false⟩` 初始值：false

title/.../fixskip

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。

`ensureskip = ⟨true|false⟩` 初始值：false

title/.../ensureskip

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。

`break = {⟨code⟩}`
`break+ = {⟨code⟩}`

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

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 标题：

例 17

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

`afterindent = ⟨true|false⟩`

title/.../afterindent

afterindent 选项用于设置章节标题后首段的缩进。

2.6.5 浮动体

CusTeX 提供了可以控制本章节内的浮动体位置的接口。

`float-barrier = ⟨true|false⟩`

title/.../float-barrier

控制是否本章节所属的浮动体可以位于其它章节内。默认情况下浮动体可以放置在其它章节内（值为 true）。

除了使用以上这个选项，还可以设置浮动体的“边界”。

阻止 \FloatBarrier 前的浮动体被放置在这个命令的后边。

\FloatBarrier

2.6.6 杂项

`tocline = {⟨格式定义⟩}`

title/.../tocline

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。

`mark = {⟨mark code⟩}`

title/.../mark

写入标记文本。⟨mark code⟩ 其后跟一个参数，在 \chapter 中，为 \chaptermark，在 \section 中，为 \sectionmark。初始情况下，不写入标记。

`bookmark = {⟨text⟩}`
`bookmark* = {⟨text⟩}`

title/.../bookmark
title/.../bookmark*

设置此条标题在书签中的文字。

bookmark* 还会在 bookmark 宏包设置了 numbered 选项后，把数字也加上。

title/.../style

```
style = {⟨style⟩}
```

设置已有的样式。

\titleifname

```
\titleifname {⟨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）。

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”。

2.6.7 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。

\enablecombinedlist

\enablecombinedlist 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。

cbl-setup/from

```
from = {⟨file⟩}                                    初始值：\jobname
```

设置目录的来源。默认为 \jobname，即来自本文件。不包含文件后缀。

cbl-setup/this
cbl-setup/write

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

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。

\tableofcontents
\listoffigures
\listoftables

输出标准的目录、图片和表格目录。

\standardplaincombinedlist
\multicolplaincombinedlist

```
\standardplaincombinedlist {⟨title⟩} {⟨cbl type⟩}
\multicolplaincombinedlist [⟨multicolumns key-val⟩] {⟨title⟩} {⟨cbl type⟩}
```

\standardplaincombinedlist 输出默认的目录，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。

\multicolplaincombinedlist 输出多栏目录，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。

仿照 etoc 宏包，提供了类似于 \etocsetstyle 的命令。

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

```
\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⟩)
```

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 用于输出局部章节的目录。

详细用法见第四章。

```
\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⟩)
```

\SetSpecifiedCombinedListStyle
\SpecifiedCombinedList
\LocalSpecifiedCombinedList

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 3.4 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。

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

\CUSProvideLibrary
\CUSProvideExplLibrary

提供库文件。库文件名必须是：“cus.library.⟨库名⟩.tex”。

```
\CUSDependency {⟨key-val⟩}
```

\CUSDependency

处理库依赖。

```
package = {⟨comma list⟩}
module  = {⟨comma list⟩}
library = {⟨comma list⟩}
disable = {⟨comma list⟩}
```

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

\CUSDependency 可用的键值选项。

```
\CUSLoadLibrary      {⟨库名⟩} [⟨日期⟩]
\CUSIfLibraryAtLeast {⟨库名⟩} {⟨日期⟩} {⟨true code⟩} {⟨false code⟩}
```

\CUSLoadLibrary
\CUSIfLibraryAtLeast

§ 1 LaTeX 2ε的钩子机制

本节简述 LaTeX 2ε的钩子机制。更详细的说明见 lthooks.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⟩}
```

\UseHook

在命令或环境中执行 ⟨hook⟩。

```
\AddToHook {⟨hook⟩} [⟨label⟩] {⟨code⟩}
```

\AddToHooks

将 ⟨code⟩ 添加到 ⟨hook⟩ 中，标记为 ⟨label⟩。⟨hook⟩ 可以未被定义。

如果未指定 ⟨label⟩，则使用默认的 label。如果 \AddToHook 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 ⟨*label*⟩ 的 ⟨*hook*⟩。若 ⟨*label*⟩ 未指定，则使用默认的 label。

\AddToHookNext
\ClearHookNext

```
\AddToHookNext {⟨hook⟩} {⟨code⟩}
\ClearHookNext {⟨hook⟩}
```

将 ⟨*code*⟩ 添加到 ⟨*hook*⟩ 的下一次调用中。在正常的 ⟨*hook*⟩ 代码执行完毕后再执行 ⟨*code*⟩。

§ 2 util 模块

3.2.1 交叉引用、超链接和书签

LaTeX 的 \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，需要用户自行加载或使用 CusTeX 提供的宏包加载机制来加载。

\lableinfo ⋆
\cus_label_info:nn ⋆

```
\lableinfo {⟨info⟩} {⟨label⟩}
```

获取 ⟨*label*⟩ 的相关信息。⟨*label*⟩ 可以为空，代表最近写入的那个 label。

可获得的信息 {⟨*info*⟩} 为：

- name，⟨*label*⟩ 的值；
- page，获得 ⟨*label*⟩ 所在页的 \thepage 值，若 ⟨*label*⟩ 不存在则为 0；
- data，获得 ⟨*label*⟩ 中的第一个数据项，若 ⟨*label*⟩ 不存在则为 \c_novalue_tl，可使用 \IfNoValueTF 或 \tl_if_novalue:nTF 判断；
- anchor，获得链接到 ⟨*label*⟩ 所在位置的锚点，若 ⟨*label*⟩ 不存在或未加载 hyperref 则为 Doc-Start；
- pageanchor，获得链接到 ⟨*label*⟩ 所在页的锚点，若 ⟨*label*⟩ 不存在或未加载 hyperref 则为 Doc-Start。

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

\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⟩} {⟨data⟩} {⟨thepage⟩}
    {⟨current label name⟩} {⟨anchor⟩} {⟨extra⟩}
```

定义一个新的 ⟨*label*⟩。当未加载 hyperref 宏包时，后三个参数无效。

⟨*label*⟩、⟨*thepage*⟩、⟨*current label name*⟩、⟨*anchor*⟩ 总是立即展开。

\cus_make_target:n
\cus_make_unique_target:n

```
\cus_make_target:n {⟨target⟩}
\cus_make_unique_target:n {⟨target⟩}
```

\cus_make_target:n 以 ⟨*target*⟩ 为名，创建一个锚点。⟨*target*⟩ 必须唯一。锚点位置自动升高 \normalbaselineskip。

\cus_make_unique_target:n 创建一个锚点，其名以 ⟨*target*⟩ 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。

```
\cus_gset_next_anchor_name:n  {⟨name⟩}
\cus_gset_next_anchor_raise:n {⟨dim⟩}
```

\g_cus_anchor_tl
\cus_gset_next_anchor_name:n
\cus_gset_next_anchor_raise:n

\g_cus_anchor_tl 保存了最近一个锚点的名称，它是只读的。相当于 \@currentHref。

\cus_gset_next_anchor_name:n 设置下一个锚点的名称。⟨name⟩ 被立刻展开。

\cus_gset_next_anchor_raise:n 设置下一个锚点要升高的高度。⟨dim⟩ 立即被计算。

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，但不可断行。

⟨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。

例 18

```
\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
```

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。

```
\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 是对 \bookmark 的封装。加载了 bookmark 宏包才有效。

\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

```
\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⟩}
```

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。⟨*page label*⟩ 一般包含 \thepage 或 \arabic{page} 等内容。\cus_gset_next_page_label:n 相当于 hyperref 宏包的 \thispdfpagelabel 命令，用于设置本页的 page label。

⟨*code*⟩ 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，⟨*code*⟩ 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。

例 19

```
\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
```

3.2.2 向前查找和收集内容

LaTeX3 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_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。

⟨*else*⟩ 的情况一般是后面的记号为 \outer 宏。

\cus_peek_box:NNnw ⟨box⟩ ⟨primitive box cs⟩ {⟨code⟩} ⟨material⟩

\cus_peek_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 \cus_peek_box:Nnw 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 20

```
\ExplSyntaxOn
\cs_set:Npn \myfbox
  {
    \cus_peek_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

\cus_peek_value:Nnw ⟨register⟩ {⟨code⟩} ⟨value⟩

\cus_peek_value:Nnw

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 21

```
\ExplSyntaxOn
\cs_set:Npn \showintval
  { \cus_peek_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 1

3.2.3 分析记号

⟨token list⟩ 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:nTF {⟨tl⟩} {⟨true code⟩} {⟨false code⟩}

\cus_if_head_int_p:n ⋆
\cus_if_head_int:nTF ⋆

测试 ⟨tl⟩ 是否以数字起始。

例 22

```
\ExplSyntaxOn
  1.  \cus_if_head_int:nTF { 2022  } { T } { F }
\ 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_split_bracket_head_default:nn ⋆
\cus_split_bracket_head:n ⋆

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

解析 ⟨*tl*⟩。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。

例 23

```
\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}]{}

\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 ⟨*tl*⟩。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。

例 24

```
\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]}[{}]

\cus_parse_range:nnnN
\cus_parse_range:(nnvN|nneN)
\cus_parse_range:nnN
\cus_parse_range:(nvN|neN)

\cus_parse_range:nnnN {⟨最小值⟩} {⟨最大值⟩} {⟨*range list*⟩} ⟨*sequence*⟩
\cus_parse_range:nnnN {⟨最大值⟩} {⟨*range list*⟩} ⟨*sequence*⟩

解析一个整数列表，将其保存至 ⟨*sequence*⟩ 中。可使用 -> 标记连续的范围。若范围的开始为空，则设它为 ⟨最小值⟩，若终止为空，则设它为 ⟨最大值⟩。

逆序的范围无效。

如果 ⟨最小值⟩ 被省略，则设它为 1。

是否检查越界值。

\cus_parse_range_check:
\cus_parse_range_nocheck:

当设置了检查越界值时，结果的项被限制在 ⟨最小值⟩ 和 ⟨最大值⟩ 之中（含边界）。

否则，忽略这一限制，但逆序的范围仍然无效。

例 25

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

\cus_parse_range:nnN { 10 } { -1->2, 4->7, 9->12, 20, 30->32 } \l_tmpa_seq
\seq_use:Nn \l_tmpa_seq { ,~ } \par

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

\cus_parse_range: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

\cus_set_parse_range_delimiter:n {⟨*delimiter*⟩}
\cus_set_parse_range_default_delimiter:

\cus_set_parse_range_delimiter:n
\cus_set_parse_range_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。

例 26

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

1, 2, 7, 8, 9, 10

3.2.4 杂项

\cus_if_preamble:TF {⟨*true code*⟩} {⟨*false code*⟩}

测试是否在导言区、正文或在 \documentclass 后。

\cus_if_preamble_p: ⋆
\cus_if_preamble:*TF* ⋆
\cus_if_document_p: ⋆
\cus_if_document:*TF* ⋆
\cus_if_after_documentclass_p: ⋆
\cus_if_after_documentclass:*TF* ⋆

\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*⟩ {⟨$tokens_1$⟩} {⟨$tokens_2$⟩}

展开 ⟨*tokens*⟩ 或 ⟨$tokens_2$⟩ 两次。

例 27

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

\CusTeX \CusTeX \CusTeX \CusTeX \CusTeX

例 28

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

\token_to_str:N

\cus_use_none_num:nw ⋆

\cus_use_none_num:nw {⟨*num*⟩} ⟨*tl*⟩

移除 ⟨*tl*⟩ 的前 ⟨*num*⟩ 项。⟨*tl*⟩ 必须至少有 ⟨*num*⟩ 项。忽略未被保护的空格。

两次展开即可得到结果。

例 29

```
\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

3.2.5 psr，处理器

有时，一个命令需要根据不同的设置显示不同的效果，但这个命令的执行逻辑已经确定，重新修改这个命令不是一个好的做法，因为无法保证对它的修改是正确的。一般可以通过在这些命令中插入钩子（hook）或修改这个命令内部真正有效的命令来间接地修改它。后者就是处理器的主要思想。

处理器是包装过的宏。

一个处理器是在某些命令内部发挥作用的接口宏，处理器的规则是处理器真正发挥执行操作的宏，通过让处理器遵循（obey）某个规则来控制处理器以何种方式运行（这些规则也是宏）。这样我们只需定义一些规则，然后根据需要让处理器遵循某个规则，从而可以达到不同的效果。

与使用条件判断相比，具有更一致的接口，可以增加任意个处理方式，并且在同一次展开时不必重复判断。

与直接使用宏相比，调用接口更加一致。

\cus_new_psr:nnn {⟨处理器⟩} {⟨参数数目⟩} {⟨code⟩}
\cus_use_psr:n {⟨处理器⟩}

\cus_new_psr:nnn
\cus_set_psr:nnn
\cus_gset_psr:nnn
\cus_use_psr:n

创建、使用处理器。

每个处理器的参数数目固定。在使用处理器时，其后需有对应数目的参数。

处理器以其名称唯一识别，不可定义参数数目不同的同名处理器。

\cus_new_psrrule:nnn {⟨处理器⟩} {⟨规则⟩} {⟨code⟩}

\cus_new_psrrule:nnn
\cus_set_psrrule:nnn
\cus_gset_psrrule:nnn

创建从属于 ⟨处理器⟩ 的规则 ⟨规则⟩。该 ⟨规则⟩ 可使用 ⟨处理器⟩ 预先定义的参数。

\cus_obey_psrrule:nn {⟨处理器⟩} {⟨规则⟩}

\cus_obey_psrrule:nn
\cus_gbey_psrrule:nn

对 ⟨处理器⟩ 局部或全局应用 ⟨规则⟩。注意，全局应用不是 gobey。

\cus_psr_if_exist:nTF {⟨处理器⟩} {⟨true code⟩} {⟨false code⟩}
\cus_psrrule_if_exist:nTF {⟨处理器⟩} {⟨规则⟩} {⟨true code⟩} {⟨false code⟩}
\cus_psr_if_compatible:nnTF {⟨处理器$_1$⟩} {⟨处理器$_2$⟩} {⟨true code⟩} {⟨false code⟩}

\cus_psr_if_exist_p:n ⋆
\cus_psr_if_exist:nTF ⋆
\cus_psrrule_if_exist_p:nn ⋆
\cus_psrrule_if_exist:nnTF ⋆
\cus_psr_if_compatible_p:nn ⋆
\cus_psr_if_compatible:nnTF ⋆

判断处理器、处理器的规则是否存在。或判断两个处理器是否兼容，当前，两个参数数目一致时视为兼容。

\cus_exec_psrrule:nn {⟨处理器⟩} {⟨规则⟩}

\cus_exec_psrrule:nn

直接执行 ⟨处理器⟩ 的 ⟨规则⟩。可用于其它规则中，相当于一个宏。其后需有对应数目的参数。

\cus_psr_argument_count:n {⟨处理器⟩}

\cus_psr_argument_count:n ⋆

计算处理器可用的参数数目。

\cus_new_psrrule_eq:nnn {⟨处理器⟩} {⟨规则$_1$⟩} {⟨规则$_2$⟩}

\cus_new_psrrule_eq:nnn
\cus_set_psrrule_eq:nnn
\cus_gset_psrrule_eq:nnn

将 ⟨规则$_1$⟩ 设置为与 ⟨规则$_2$⟩ 相等，它们从属于 {⟨处理器⟩}。

\cus_new_psrrule_eq_cs:nnN {⟨处理器⟩} {⟨规则⟩} ⟨function⟩

\cus_new_psrrule_eq_cs:nnN
\cus_new_psrrule_eq_cs:nnc
\cus_set_psrrule_eq_cs:nnN
\cus_set_psrrule_eq_cs:nnc
\cus_gset_psrrule_eq_cs:nnN
\cus_gset_psrrule_eq_cs:nnc

将 ⟨处理器⟩ 的 ⟨规则⟩ 设置为与 ⟨function⟩ 相等。这 ⟨function⟩ 的参数数必须与 ⟨处理器⟩ 的参数数相等，且必须是非定界的变量（undelimited parameter）。

§ 3 box 模块

box 模块封装了一些环境或命令，用于在编程时使用。

\cus_varwidth:nnw {⟨vertical pos⟩} {⟨maximum width⟩} ⟨contents⟩ \cus_varwidth_end:

\cus_varwidth:nnw
\cus_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。

\cus_set_varwidth:Nnnw ⟨box⟩ {⟨vertical pos⟩} {⟨maximum width⟩}
⟨contents⟩ \cus_set_varwidth_end:

\cus_set_varwidth:Nnnw
\cus_set_varwidth_end:

设置 ⟨box⟩ 为一个包含 ⟨contents⟩ 的最大宽度为 ⟨maximum width⟩ 的水平盒子。

\cus_set_vbox_width:Nnw
\cus_set_vbox_width_end:
\cus_set_vbox_varwidth:Nnw
\cus_set_vbox_varwidth_end:

\cus_set_vbox_width:Nnw ⟨*box*⟩ {⟨*width*⟩} ⟨*content*⟩ \cus_set_vbox_width_end:

\cus_set_vbox_width:Nnw 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。

\cus_set_vbox_varwidth:Nnw 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 \vbox_set_split_to_ht:NNn 来分割。

3.3.1 收集宽度固定和宽度可变的内容

util 模块提供了收集内容的命令，本模块则进一步提供了收集宽度固定和宽度可变的内容的命令。

\cus_peek_width:Nnnnw
\cus_peek_minipage:Nnnnw
\cus_peek_varwidth:Nnnnw

\cus_peek_minipage: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，默认为居中。

⟨*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。

\cus_peek_width:Nnnnw 是 \cus_peek_minipage:Nnnnw 的另一个名字。

例 30

```
\ExplSyntaxOn
\cs_set:Npn \myparfbox #1#2
  {
    \cus_peek_minipage:Nnnnw \l_tmpa_box
      { \fbox { \box_use_drop:N \l_tmpa_box } } {#1} {#2}
  }
\cs_set:Npn \myvarfbox #1#2
  {
    \cus_peek_varwidth:Nnnnw \l_tmpa_box
      { \fbox { \box_use_drop:N \l_tmpa_box } } {#1} {#2}
  }
\ExplSyntaxOff
\myparfbox{t}{5cm}{一个可以包含 \verb|\verb| 的\par
  定宽 \verb|\fbox|}
\myvarfbox{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

\cus_peek_width:Nnnnnnw
\cus_peek_minipage:Nnnnnnw
\cus_peek_varwidth:Nnnnnnw

`\cus_peek_minipage:Nnnnnnw` ⟨*box*⟩ {⟨*code*⟩}
{⟨*vpos*⟩} {⟨*height*⟩} {⟨*inner pos*⟩} {⟨*width*⟩} ⟨*material*⟩

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，默认为居中对齐；
- ⟨*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。

3.3.2 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`、`\cus_ref_target_box:nn`。

例 31

```
\ExplSyntaxOn
\cs_set:Npn \myparfbox
  {
    \cus_peek_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
本节开始 |链接到 | 链接到
本节开始

另见 `\HyperRef`、`\HyperLink` 及例 43。

§ 4 struct 模块

以下简单描述 struct 模块中目录的数据结构。

\addcombinedlisttype

\addcombinedlisttype {⟨type⟩} {⟨cbl levels⟩}

若要添加新的目录类型，必须先声明 ⟨type⟩。⟨cbl levels⟩ 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 ⟨level⟩，也可使用通常的 key=val 的形式。

比如，对于标准的目录 \tableofcontents，它写入的 type 为 toc，有

例 32
```
\addcombinedlisttype{toc}
  {
    part[-1],
    chapter[0],
    section[1], subsection[2], subsubsection[3],
    sub3section[4], sub4section[5],
    paragraph[4], subparagraph[5],
  }
```

对于标准的 \listoffigures 和 \listoftables，有

例 33
```
\addcombinedlisttype{lof}{ figure=1 }
\addcombinedlisttype{lot}{ table=1 }
```

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 ⟨type⟩，其第二个参数就是这里的 ⟨cbl level⟩，即层级名称，⟨cbl levels⟩ 应包含这个参数的所有可能值；第三个参数就是 ⟨list entry⟩。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。

\retcbltypelevel ⋆

\retcbltypelevel {⟨type⟩} {⟨cbl level⟩}

展开为 ⟨type⟩ 类型中层级名称 ⟨cbl level⟩ 对应的层级数。前缀 ret 为 return。

\retcbltotalcounts ⋆

\retcbltotalcounts {⟨type⟩}

展开为 ⟨type⟩ 类型的目录条目数。若 ⟨type⟩ 为空，则为各类型的总和。

每个类型的条目数在使用 \enablecombinedlist 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 {⟨type⟩}{⟨cbl count⟩}{⟨tags⟩}{⟨level⟩}
{⟨list entry⟩}{⟨thepage⟩}{⟨anchor⟩}\cus@type@contentsline@

- \cus@type@contentsline，\cus@type@contentsline@，这两个宏标记条目的边界；
- ⟨type⟩ 为目录类型名；
- ⟨cbl count⟩ 为本条目在存储着所有目录项的那个列表中的位置；
- ⟨tags⟩ 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。在 \chapter 中为 chapter，在 figure 中，为 figure；
- ⟨level⟩ 为这个目录项的层级，是一个整数；
- ⟨list entry⟩ 为这个目录项的值，通常包含着标题或 caption；
- ⟨thepage⟩ 为目录项所在页的 \thepage；
- ⟨anchor⟩ 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@、⟨*cbl count*⟩ 改为 \cus@cbl@contentsline、\cus@cbl@contentsline@、⟨*type count*⟩ 外，其它未改变。这里的 ⟨*type count*⟩ 为此条目在其对应类型的列表中的位置。

\retcblentryname {⟨*type*⟩} {⟨*count*⟩}

\retcblentryname ⋆

展开为存储着类型 ⟨*type*⟩ 第 ⟨*count*⟩ 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 \tl_show:c {\retcblentryname{}{\retcbltotalcounts{}}} 在终端中显示 cbl 列表的最后一项。

\retcblentrydata {⟨*type*⟩} {⟨*data*⟩} {⟨*count*⟩}

\retcblentrydata ⋆

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、tags、level、entry、thepage、anchor。

\iteratecontents {⟨*type*⟩} {⟨*inline code*⟩}

\iteratecontents

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*⟩}

\retcbldefaultlevellistname ⋆

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。

\CurrentCombinedListCount

当写入目录时这个 int 寄存器递增一次。

在此处，它的值为 46，表示此前最近一次写入的目录是目录文件中的第 46 项。

\CurrentTocDefaultLevelCount

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 \chapter 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。

例 34

```
\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
```

29, 42。表示本章节的所有目录项为 toc 类型的目录中的第 28 – 40 项。

以下代码输出本章目录。

例 35

```
\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
```


`\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。

§ 5 LaTeX 2ε的 mark 机制

LaTeX 2ε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 {⟨class⟩} {⟨text⟩}
\mark_insert:nn {⟨class⟩} {⟨text⟩}
```

```
\InsertMark
\mark_insert:nn
insertmark
```

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 multocols 等多栏环境中使用它们将无效。

在执行 \InsertMark、\mark_insert:nn 时，将首先执行 insertmark 钩子。

```
\TopMark [⟨region⟩] {⟨class⟩}
\mark_use_last:nn {⟨region⟩} {⟨class⟩}
```

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

展开为 ⟨*class*⟩ 在 ⟨*region*⟩ 中相应位置的 ⟨*text*⟩。

\FirstMark、\LastMark 分别展开为 ⟨*region*⟩ 的第一个、最后一个 ⟨*text*⟩。

\TopMark 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。

```
\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

CuSTEX 重新实现了标题和目录的命令。其中标题的设置方式是 CTEX 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

§ 2　输出 LATEX 原始风格的目录

CuSTEX 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 \enablecombinedlist 命令启用。这个命令只能使用一次，可以用于导言区（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 \tableofcontents 等命令来输出目录。

默认情况下，使用 \tableofcontents、\listoffigures、\listoftables 就是 LATEX 默认的格式。这些命令是对 \standardplaincombinedlist 的简单封装，即

例 36
```
\renewcommand{\tableofcontents}{%
  \standardplaincombinedlist{\contentsname}{toc}}
\renewcommand{\listoffigures}{%
  \standardplaincombinedlist{\listfigurename}{lof}}
\renewcommand{\listoftables}{%
  \standardplaincombinedlist{\listtablename}{lot}}
```

\standardplaincombinedlist 的第一个参数是显示的标题名称，第二个参数是目录类型。

\multicolplaincombinedlist 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 2.3.3 小节，随后的两个参数和 \standardplaincombinedlist 一样。

CusTeX 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。

宏包	*type*	*level name*	宏包	*type*	*level name*
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

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 \section 同级。可以使用 \addcombinedlisttype 修改。

§ 3 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 和 \specifiedlot 来分别设置和输出目录，它们独立于 \tableofcontents 命令，互不干扰。对于图片和表格也有类似的命令：

\lofsetstyle、\specifiedlof、\lotsetstyle、\specifiedlot。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：

例 37
```
\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]}
```

因此，只需介绍 \SetSpecifiedCombinedListStyle 和 \SpecifiedCombinedList 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*。

\SpecifiedCombinedList 接受一个可选参数 ⟨*type*⟩，即目录类型。用于输出这个目录类型的目录条目。\SetSpecifiedCombinedListStyle 有一个可选参数和 6 个必需参数。分别为：

1. ⟨*type list*⟩，目录类型的列表；
2. ⟨*level list*⟩，层级列表，由 *level name* 或数字组成的列表；
3. ⟨*list start*⟩，此层级列表开始时执行的代码；
4. ⟨*block start*⟩，此层级的块开始时执行的代码；
5. ⟨*block item*⟩，此条目执行的代码；
6. ⟨*block finish*⟩，此层级的块结束时执行的代码；
7. ⟨*list finish*⟩，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 39。

在这些参数中可以，定义了如下的命令来辅助制作目录：

\tocthenumber 标题前的数字。例如：“第一章”、“§ 1”；可能为空，可以使用 \tocifnumbered 命令判断；

\tocthename 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；

\tocifnumbered {⟨*true*⟩}{⟨*false*⟩}，判断该目录条目是否有编号；

\tocifpageisnumber {⟨*true*⟩}{⟨*false*⟩}，判断页码是否为数字；

\toclink {⟨*text*⟩}，创建超链接，把 ⟨*text*⟩ 链接到文档中的对应位置；

\toclinkbox {⟨*context*⟩}，同上，对于非文字内容也能正确跳转；

\tociffirst {⟨*true*⟩}{⟨*false*⟩}，判断当前目录条目是否是其所在的 *level list* 的第一项；

\tocifheadentry {⟨*true*⟩}{⟨*false*⟩}，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 {⟨*true*⟩}{⟨*false*⟩}，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、\HyperLink 等命令的第一个参数；

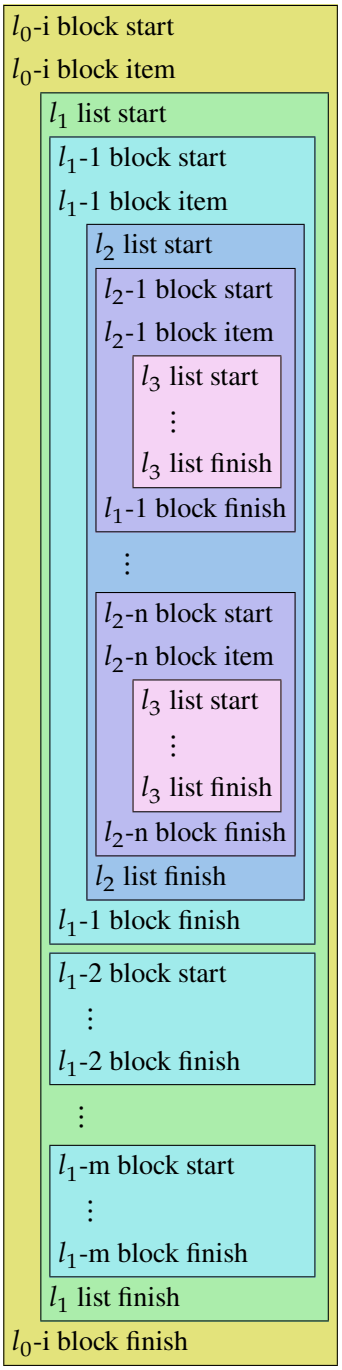


图 4.1: specified cbl 层级图

\tocthelevel 此条目的层级；
\tocthеprevlevel 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*；
\tocthеindex 表示该条目在所属的目录类型中是第几项；
\tocifcomplement {⟨*true*⟩}{⟨*code*⟩}，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；
\tocthetags 目录条目中可能包含的某些特殊信息；
\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 的强大能力（尽管它是用 \tocsetstyle 设置的）。结果如图 4.2 所示。

例 38

```
\newcommand{\ifinmiddle}[2]{\ifnum\tocthelevel=\tocthenextlevel\relax
↪ #1\else #2\fi}
% \usepackage{enumitem}
% \setlist[description]{nosep}
\tocsetstyle
  {chapter,section}
  {\begin{description}}
  {}
  {\item[\tocifnumbered{\tocthenumber}{\rule{1ex}{1ex}}]
    \tocthename\quad\toclink{\tocthepage}\par}
  {}
  {\end{description}}
\tocsetstyle{subsection}
  {\par\begingroup\small\itshape\raggedright【\ }
  {}
  {\tocthenumber\enskip\tocthename
     (\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

图 4.2: 例 38 的结果

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。

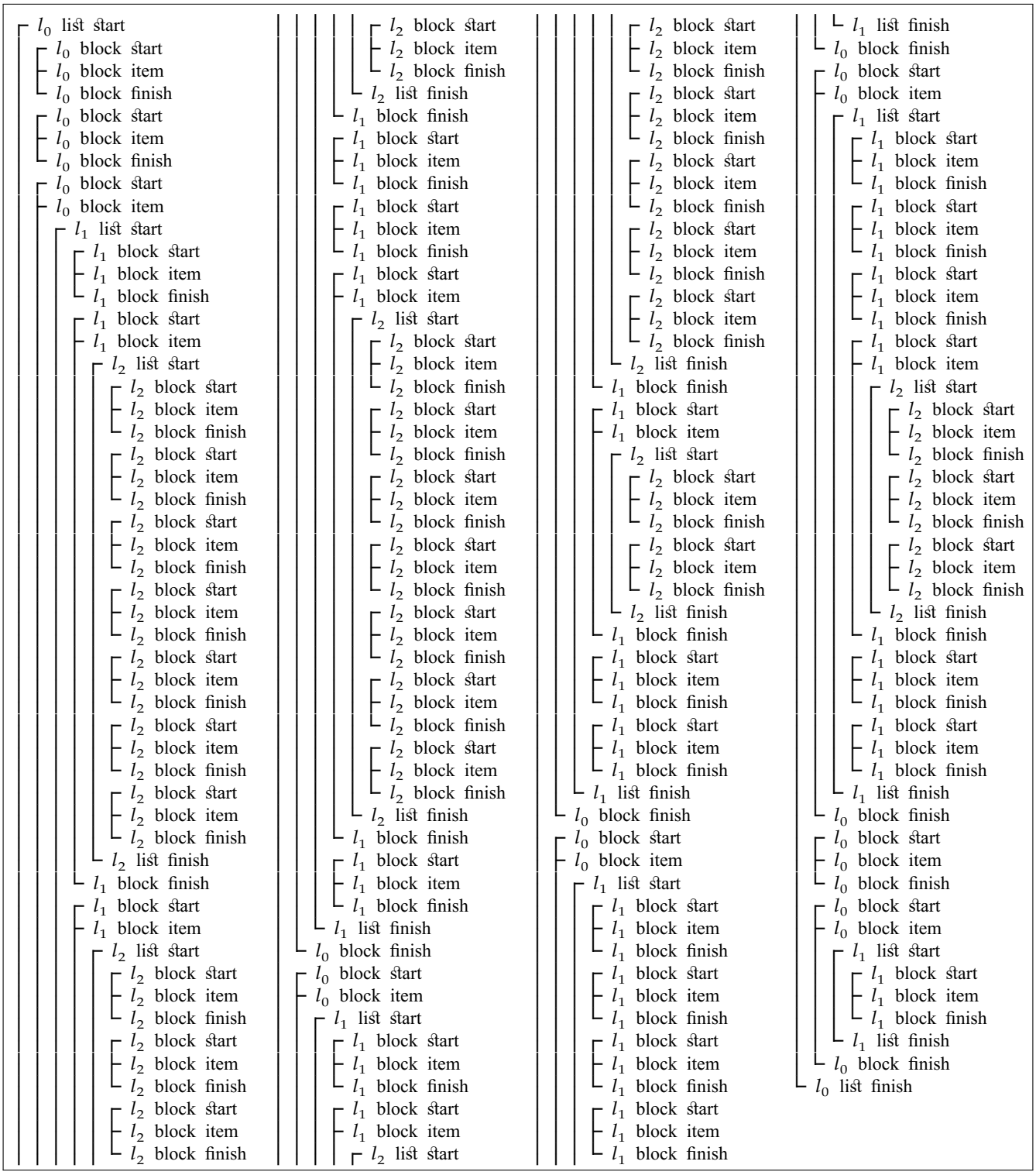
想要双栏并排显示只需使用 \startmulticolumns 即可。

下面这个例子展示了本文目录在 CuSTEX 目录机制下的输出顺序。

例 39

```
\startmulticolumns[cols=4,ragged,column-sep=10pt]
\newcommand\showthelevel{$l_{\tocthelevel}$ }
\newcommand\showhbarl{\Replicate{2*(\tocthelevel-\tochilevel)}{|}}
\newcommand\showhbarb{\Replicate{2*(\tocthelevel-\tochilevel)+1}{|}}
\tocsetstyle{chapter,section,subsection,0,1,2}
  {\showhbarl ┌ \showthelevel list start\par}
  {\showhbarb ┌ \showthelevel block start\par}
  {\showhbarb ├ \showthelevel block item\par}
  {\showhbarb └ \showthelevel block finish\par}
  {\showhbarl └ \showthelevel list finish\par}
\setlength{\parindent}{0pt}
\setlength{\lineskip}{0pt} \setlength{\lineskiplimit}{\maxdimen}
```

```
\fontspec{TH-Times}\small
\specifiedtoc
\stopmulticolumns
```



下例展示了为目录添加盒子和边距的方法：

例 40

```
\startmulticolumns[2,ragged,outer-sep=0pt]

\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 \tocifnumbered{\tocthenumber\quad\kern-.3em}{}\tocthename
      \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}
  {\tocifnumbered{\tocthenumber\quad}{}\tocthename\breakablefiller[space]%
    \makebox[2em][r]{\tochyperpage\;}\par }
  {\endgroup}
  {\end{list}}
\tocsetstyle {subsection,2}
  {}
  {\begingroup\color{black}\bfseries}
  {\tocifnumbered{\tocthenumber\quad}{}\tocthename\breakablefiller[dotted]%
    \makebox[2em][r]{\tochyperpage\;}\par }
  {\endgroup}
  {}
\makeatother

\specifiedtoc

\stopmulticolumns
```

还可使用 \LocalSpecifiedCombinedList 来输出局部目录。\localspecifiedtoc 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 \section 中，输出的局部目录为此 \section 小节的目录。但可以设置 \localspecifiedtoc(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 \localspecifiedtoc(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 \CurrentCombinedListCount。

\LocalSpecifiedCombinedList 目前还存在一点问题。**[TODO]**

关于局部章节目录，另见例 35。

§ 4 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 \addcombinedlisttype 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 LaTeX 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，\part 为 −1，\chapter 为 0，\section 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 \part、\chapter、\section、\subsection、\subsubsection、\paragraph、\subparagraph 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，LaTeX 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　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。

本库的详细用法可参考本说明文档的源码。

本库提供 \cs、\cmd、\tn 来排版宏。

```
\cs  [⟨cmd key-val⟩] {⟨macro name⟩}
\cmd [⟨cmd key-val⟩] {⟨macro⟩}
\tn  [⟨cmd key-val⟩] {⟨tex macro name⟩}
\key [⟨cmd key-val⟩] {⟨key name⟩}
```

```
\cs
\cmd
\tn
\key
```

排版宏。\tn 专用于排版 TeX 和 LaTeX 2ε的宏。

```
index    = {⟨index entry⟩}
module   = {⟨module name⟩}
no-index = ⟨true|false⟩                          初始值：false
do-index = ⟨true|false⟩
space    = ⟨true|false⟩                          初始值：false
```

```
doc/cmd/index
doc/cmd/module
doc/cmd/no-index
doc/cmd/space
```

\cs、\cmd、\tn 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 function，keyval 和 syntax 环境中时，不写入索引文件。

space 控制是否将空格替换为 \textvisiblespace。

\meta
\veta
\Arg
\oarg
\parg
\pkg
\env
\cls
\opt
\file
\docfile

排版宏的参数。

function

```
\begin{function} [⟨function key-val⟩] {⟨functions clist⟩}
  ...
\end{function}
```

显示函数说明。⟨*functions clist*⟩ 可以是宏或者环境名。

keyval

```
\begin{keyval} [⟨function key-val⟩] {⟨keys clist⟩}
  ...
\end{keyval}
```

显示键值选项的说明。⟨*keys clist*⟩ 为键列表。

syntax

```
\begin{syntax}
  ...
\end{syntax}
```

输出使用方法。

本环境中每个输入行都为一个输出行（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{...}。

TEXhackers note: # 在本环境中的类别码被设置为 12（other）。

function，keyval 和 syntax 环境均可使用 \V 命令，它和 \Verbatimize 一样，但以当前字体显示。

EXP 不可设置值
rEXP 不可设置值

doc/function/EXP
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
pTF 不可设置值
noTF 不可设置值

doc/function/TF
doc/function/pTF
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:nn。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⟩}
updated = {⟨年⟩-⟨月⟩-⟨日⟩} 或 {⟨年⟩/⟨月⟩/⟨日⟩}

doc/function/added
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*⟩}
label* = {⟨*label list*⟩}
no-label 不可设置值

doc/function/label
doc/function/label*
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。

frame = {⟨*frame key-val*⟩}
frame+ = {⟨*frame key-val*⟩}

doc/function/frame
doc/function/frame+

设置外部方框盒子的选项。

```
\begin{texnote}
  ...
\end{texnote}
```

texnote

\csref
\csreflist
\envref
\envreflist
\keyref
\keyreflist

```
\csref       [⟨类型⟩] {⟨cs name⟩}
\csref       [⟨类型⟩] (⟨cs label⟩) {⟨cs name⟩}
\csreflist   [⟨类型⟩] {⟨cs name list⟩}
\envref      [⟨类型⟩] {⟨env name⟩}
\envref      [⟨类型⟩] (⟨cs label⟩) {⟨env name⟩}
\envreflist  [⟨类型⟩] {⟨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@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，\envref 和 \keyref 命令中链接的颜色。可以使用 \colorlet 等命令修改。

§ 2 bnf 库

bnf 库用于排版基于 Backus-Naur Form（BNF 范式）的文法。

latexbnf

```
\begin{latexbnf}[⟨texbnf key-val⟩]
  <⟨non-terminal⟩> : <⟨non-terminal⟩>
  <⟨non-terminal⟩> : "⟨terminal⟩"
  <⟨non-terminal⟩> : <⟨non-terminal⟩> | <⟨non-terminal⟩>
  ...
\end{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 \lo、\hi，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 \Verbatimize。

\BNFN {⟨*non-terminal*⟩}
\BNFT {⟨*terminal*⟩}

\BNFIItem
\BNFN
\BNFI
\BNFO
\BNFT

\BNFItem 用于标记一个句法（syntax）的开始。

\BNFN 排版非终结符。\BNFT 排版终结符，它的使用方式和效果与 \verb 类似。

\BNFI 表示它之前的内容被定义为它之后的内容。

\BNFO 表示“或者”。

除 \BNFItem 外，上述命令均可在正文环境中使用。

在 latexbnf 环境中，可使用 \is 代替 \BNFI，\alt 代替 \BNFO，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⟩ 初始值：**false**
hyper-color = {⟨颜色⟩}

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。

\BNFN 超链接的使用与否也受 hyper 选项控制。

例 41

```
\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 42

```
\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*⟩} 初始值：**\ttfamily**
clear-all-format 不可设置值

设置格式。

texbnf/Nleft
texbnf/Nright
texbnf/Tleft
texbnf/Tright
texbnf/N
texbnf/T
texbnf/O
texbnf/I

Nleft = {⟨*code*⟩} 初始值：**\ensuremath{\langle}**, ⟨
Nright = {⟨*code*⟩} 初始值：**\ensuremath{\rangle}**, ⟩
I = {⟨*code*⟩} 初始值：**\ensuremath{\longrightarrow}**, ⟶
O = {⟨*code*⟩} 初始值：**\cus@mathrule**, |

设置 \BNFN、\BNFT 左右的符号，\BNFI、\BNFO 的替换符号。

texbnf/label-prefix
texbnf/label-suffix

label-prefix = {⟨前缀⟩} 初始值：**texbnf//**
label-suffix = {⟨后缀⟩}

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。

§ 3 ref 库

ref 库提供交叉引用的一些额外功能。

本库改进了 \ifPageOdd 和 \ifAbsPageOdd，使得它们在任何位置都有效。

\ifLabelOdd

\ifLabelOdd {⟨*label*⟩} {⟨*true*⟩} {⟨*false*⟩}

判断这个 ⟨*label*⟩ 是否定义在奇数页。当 ⟨*label*⟩ 不存在时使用 ⟨*false*⟩。

⟨*label*⟩ 所在页码必须以阿拉伯数字显示，否则使用 ⟨*false*⟩ 分支。

```
\HyperRef   {⟨label⟩} {⟨text⟩}
\HyperRef * {⟨label⟩} ⟨material⟩
\HyperLink   {⟨target⟩} {⟨text⟩}
\HyperLink * {⟨target⟩} ⟨material⟩
```

\HyperRef
\HyperLink

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 和 \HyperLink 用于解决这个问题。

\HyperRef{⟨*label*⟩}{⟨*text*⟩} 和 \hyperref[⟨*label*⟩]{⟨*text*⟩} 完全一样。其作用是把 ⟨*text*⟩ 链接到 ⟨*label*⟩ 所在的那个地方。

\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。

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，比如 verbatim，parbox 等，但 ⟨*material*⟩ 本身并不能作为其它命令的参数。缺点是不能换行。

另见 \cus_ref_label_box:nn、\cus_ref_target_box:nn。

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   {⟨label⟩} [⟨width⟩] ⟨material⟩
\HyperRef * {⟨label⟩} [⟨width⟩] ⟨material⟩
\HyperRef   {⟨label⟩} [⟨width⟩] [⟨vpos⟩] [⟨height⟩] [⟨inner pos⟩] ⟨material⟩
\HyperRef * {⟨label⟩} [⟨width⟩] [⟨vpos⟩] [⟨height⟩] [⟨inner pos⟩] ⟨material⟩
```

\HyperRef
\HyperLink

当不带星号使用时，相当于把 ⟨*material*⟩ 放在 minipage 中，⟨*material*⟩ 的宽度设置为 ⟨*width*⟩，但可以包含特殊文本。当带星号使用时，相当于把 ⟨*material*⟩ 放在 varwidth 中，⟨*material*⟩ 的最大宽度为 ⟨*width*⟩，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：

- ⟨*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，默认为居中对齐；
- ⟨*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。

另见 \cus_peek_minipage:Nnnnnnw、\cus_peek_varwidth:Nnnnnnw。

虽然这两个命令不能配置 ⟨*material*⟩，但 box 库提供了 \fparbox 和 \fvarbox 命令可以设置盒子外框，见第 5.4.3 小节及例 43。

§ 4 box 库

box 库提供额外的一些盒子。

5.4.1 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 ⟨*col*⟩ 栏，栏以 0 开始计数。若不给出 ⟨*col*⟩ 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 ⟨*heading*⟩ 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
```

⟨*cols*⟩ 设置多栏的总栏数。paracol 环境支持多栏分布在左右两页，通过 ⟨*numleft*⟩ 设置在左侧页的栏数。在此情况下，⟨*paired*⟩ 用于设置这左右两页的页码是否相同。

paracol/heading

```
heading = {⟨text⟩}
```

在多栏文本前插入通栏文字 ⟨*text*⟩。

paracol/column-ratio
paracol/column-ratio-left
paracol/column-ratio-right

```
column-ratio       = {⟨r0, r1, ⋯, rk⟩} [⟨r0', r1', ⋯, rk'⟩]
column-ratio-left  = {⟨r0, r1, ⋯, rk⟩}
column-ratio-right = {⟨r0', r1', ⋯, rk'⟩}
```

设置每栏宽度的占比。⟨*column-ratio*⟩ 的可选参数和 ⟨*column-ratio-right*⟩ 设置的是右侧页的栏。

每栏的实际宽度为 $r_i \times (\texttt{\textwidth} - (n-1)\texttt{\columnsep})$。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。

paracol/column-width
paracol/column-width-left
paracol/column-width-right

```
column-width       = {⟨s0, s1, ⋯, sk⟩} [⟨s0', s1', ⋯, sk'⟩]
column-width-left  = {⟨s0, s1, ⋯, sk⟩}
column-width-right = {⟨s0', s1', ⋯, sk'⟩}
```

设置每栏的宽度和间距。⟨*column-width*⟩ 的可选参数和 ⟨*column-width-right*⟩ 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 $\hat{w}_i$ =\fill，$\hat{g}_i$ =\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/20,50pt 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。

marginpar-threshold = {⟨k⟩} [⟨k'⟩]
marginpar-threshold-left = {⟨k⟩}
marginpar-threshold-right = {⟨k'⟩}

paracol/marginpar-threshold
paracol/marginpar-threshold-left
paracol/marginpar-threshold-right

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。

counter-global = {⟨*counter list*⟩} | *
counter-local = {⟨*counter list*⟩}

paracol/counter-global
paracol/counter-local

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 {⟨*counter*⟩} {⟨*col*⟩} {⟨*rep*⟩}

\definethecounter

将第 ⟨*col*⟩ 栏的 \the⟨*counter*⟩ 修改为 ⟨*rep*⟩。

\synccounter {⟨*counter*⟩}
\syncallcounter

\synccounter
\syncallcounter

将计数器 ⟨*counter*⟩ （或所有计数器）在此栏的值同步到其它栏。

column-sep-rule = {⟨长度⟩}

paracol/column-sep-rule

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 ⟨*col*⟩ 栏开始时执行的代码。⟨*col*⟩ 为 −1 表示通栏文字执行前执行的代码。如果 ⟨*col*⟩ 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。\normalcolumncolor、\normalcolseprulecolor 用于恢复为原始颜色。

如果在环境外使用，则 ⟨*col*⟩ 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 ⟨*col*⟩ 栏的章节添加到目录文件 ⟨*file*⟩。⟨*file*⟩ 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。

5.4.2 multicolumns/framed=lfbox

本库提供为多栏文字的外框提供了了 lfbox 可选值。表示用 longfbox 宏包的 \lfbox 作为盒子外框。multicolumns/framed-option 可以使用 lfbox 宏包提供的选项，但不能改变盒子的宽高和对齐方式。

5.4.3 \fparbox 和 \fvarbox，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 \fparbox 和 \fvarbox 以补全这一缺点。

\fparbox
\fvarbox

```
\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 把 ⟨*material*⟩ 封装进 minipage 中，\fvarbox 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 \HyperRef 命令的说明中。⟨*longfbox options*⟩ 为 longfbox 中可用的键值选项。

例 43

```
\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|}}
```

链接到 可以分段的 \lfbox

可以分段的
\lfbox

§ 5 math 库

\delsizel {⟨real⟩} ⟨left⟩
\delsizer {⟨real⟩} ⟨right⟩

\delsizel
\delsizer
\bigsizel
\bigsizer

正如 \bigl 和 \bigr 那样，但如上四个命令可以设置括号的大小。

\delsizel、\delsizer 设置的 ⟨real⟩ 为数学模式下左括号（ 高度的倍数。\bigsizel、\bigsizer 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 \bigsizel{1}。

§ 6 counter 库

\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。

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；
- [ ] 适配未来 LaTeX2023 的更新（latex-lab new or、footnotes）；
- [ ] 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；
- [ ] TeX、LaTeX 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。

D

M

S

T

U

V

Z