xcoffins 宏包

设计级别的排版盒子

LaTeX 项目* 【著】

黄 旭 华 † 【译】

分发于 2024-01-04

目 录

*E-mail: latex-team@latex-project.org

†一名业余 LaTeX 爱好者。

1 介绍：coffin 的概念

在 LaTeX3 术语 (terminology) 中，“coffin (排版盒子)”是一个装有排版材料 (typeset material) 的盒子 (box)。除了盒子本身，coffin 的结构 (structure) 还包括盒子尺寸 (size) 和形状 (shape) 的信息，这使得可以轻松地对齐两个或多个 coffins。这是通过为每个 coffin 提供一系列“杆 (pole)”来实现的。这些杆 (pole) 是在确定的位置 (defined positions) 如顶部或水平中心 (horizontal centre) 穿过 coffin 的水平线 (horizontal lines) 和垂直线 (vertical lines)，这些杆 (pole) 相交 (intersect) 的点称为“handles (手柄)”。然后，通过描述一个 coffin 上的手柄与另一个 coffin 的手柄之间的关系，就可以将两个 coffin 对齐。用文字表达 (in words)，一个例子可能是这样的：

> *将 coffin A 的左上手柄 (top-left handle) 与 coffin B 的右下手柄 (bottom-right handle) 对齐。*

coffin 手柄的位置从视觉上很容易理解。图 1 显示了水平模式 (horizontal mode)(左) 和垂直模式 (vertical mode)(右) 中 coffin 排版的标准手柄位置 (standard handle positions)。请注意，后一种情况会有更多的手柄可用。如图所示，每个手柄都是两个杆 (pole) 相交的结果。例如，coffin 的中心标记为`(hc,vc)`，即水平中心杆 (horizontal centre pole) 与垂直中心杆 (vertical centre pole) 的交点。当杆添加到 coffin 中时，会自动生成新的手柄：手柄是“动态的 (dynamic)”实体 (entities)。

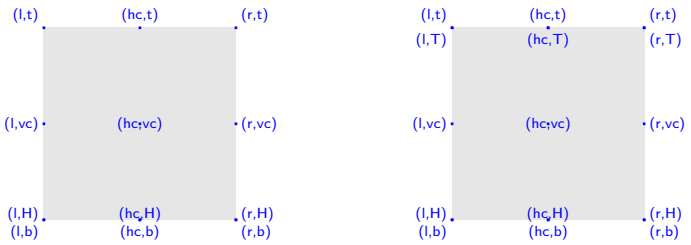

图 1: 标准的 coffin 手柄。左图，水平 coffin；右图，垂直 coffin。

2 创建和设置 coffins

在进行任何对齐 (alignment) 之前，必须创建 coffins 并创建其内容。除了 coffins 制作以外，所有的 coffins 操作 (operations) 都是当前 TEX 组 (group) 的局部操作。

\NewCoffin `\NewCoffin` ⟨*coffin*⟩

在使用 ⟨*coffin*⟩ 之前，它必须使用 `\NewCoffin` 进行分配 (allocated)。⟨*coffin*⟩ 的名称应该是一个控制序列 (control sequence) (通常以转义字符 `\` 开始)，例如：

```
\NewCoffin\MyCoffin
```

coffins 是全局分配的 (allocated globally)，如果 ⟨*coffin*⟩ 的名称不是全局唯一的 (globally-unique)，就会引起错误。

\SetHorizontalCoffin `\SetHorizontalCoffin` ⟨*coffin*⟩ {⟨*material*⟩}

在水平模式 (horizontal mode) 中排版 ⟨*material*⟩，将结果存储在 ⟨*coffin*⟩ 中。然后根据排版材料 (typeset material) 的尺寸 (size) 建立 ⟨*coffin*⟩ 的标准杆 (standard poles)。

\SetVerticalCoffin `\SetVerticalCoffin` ⟨*coffin*⟩ {⟨*width*⟩} {⟨*material*⟩}

在给定 ⟨*width*⟩ 的垂直模式 (vertical mode) 中排版 ⟨*material*⟩，并将结果存储在 ⟨*coffin*⟩ 中。然后根据排版材料 (typeset material) 的尺寸 (size) 建立 ⟨*coffin*⟩ 的标准杆 (standard poles)。

3 控制 coffin 的杆

设置或对齐 coffin 时，会自动生成多个标准杆 (standard poles)。coffins 的所有标准杆如下：

`l` 沿着 coffin 边框左侧边缘 (left-hand edge) 延伸的杆；

`hc` 垂直穿过 coffin 中心的杆，位于边框的左右边缘之间，即“水平中心”；

`r` 沿着 coffin 边框右侧边缘 (right-hand edge) 延伸的杆；

`b` 沿着 coffin 边框底部边缘 (bottom edge) 延伸的杆；

`vc` 水平穿过 coffin 中心的杆，位于边框的顶底边缘之间，即“垂直中心”；

`t` 沿着 coffin 边框顶部边缘 (bottom edge) 延伸的杆；

`H` 沿着 coffin 中排版材料的基线 (baseline) 延伸的杆。

此外，装有垂直模式材料 (vertical-mode material) 的 coffins 还具有反映这些系统更丰富性质 (nature) 的杆：

B 沿着 coffin 底部材料的基线延伸的杆。

T 沿着 coffin 顶部材料的基线延伸的杆。

\SetHorizontalPole `\SetHorizontalPole` ⟨*coffin*⟩ {⟨*pole*⟩} {⟨*offset*⟩}

设置横向穿过 ⟨*coffin*⟩ 的 ⟨*pole*⟩。⟨*pole*⟩ 位于 ⟨*coffin*⟩ 基线的 ⟨*offset*⟩ 上。这个 ⟨*offset*⟩ 应作为尺寸表达式 (dimension expression) 给出，这可能包括术语 `\TotalHeight`、`\Height`、`\Depth` 和 `\Width`，它们将计算 ⟨*coffin*⟩ 的适当尺寸 (appropriate dimensions)。例如，要在 coffin 底部到顶部距离的三分之一处创建一根横穿 coffin 的杆，适当的指令 (instruction) 应该是：

```
\SetHorizontalPole \MyCoffin {height/3} {\TotalHeight/3}
```

注意，水平 (*horizontally*) 延伸的杆是根据其在 coffin 中的垂直 (*vertical*) 位置来描述的。还请注意，coffin 的总高度 (total height) 由 `\Height` 和 `\Depth` 之和表示：这两个值都是从 coffin 中材料的水平基线 (horizontal baseline) 来测量的。

\SetVerticalPole `\SetVerticalPole` ⟨*coffin*⟩ {⟨*pole*⟩} {⟨*offset*⟩}

设置垂直穿过 ⟨*coffin*⟩ 的 ⟨*pole*⟩。⟨*pole*⟩ 位于 ⟨*coffin*⟩ 盒子边框左侧边缘 (left-hand edge) 的 ⟨*offset*⟩ 上。这个 ⟨*offset*⟩ 应作为尺寸表达式 (dimension expression) 给出，这可能包括术语 `\TotalHeight`、`\Height`、`\Depth` 和 `\Width`，它们将计算 ⟨*coffin*⟩ 的适当尺寸 (appropriate dimensions)。例如，要在 coffin 的左侧边缘三分之一处创建一根垂直穿过 coffin 的杆，适当的指令 (instruction) 应该是：

```
\SetVerticalPole \MyCoffin {width/3} {\Width/3}
```

注意，垂直 (*vertically*) 延伸的杆是根据其在 coffin 中的水平 (*horizontal*) 位置来描述的。

\TotalHeight `\TotalHeight`

在 `\SetHorizontalPole` 和 `\SetVerticalPole` 的 ⟨*offset*⟩ 参数中，`\TotalHeight` 将给出从底部 (base) 到相关 coffin 的边界框 (bounding box) 顶部 (top) 的距离。

\Height `\Height`

在 `\SetHorizontalPole` 和 `\SetVerticalPole` 的 ⟨*offset*⟩ 参数中，`\Height` 将给出从基线 (baseline) 到相关 coffin 的边界框 (bounding box) 顶部 (top) 的距离。

\Depth \Depth

在 \SetHorizontalPole 和 \SetVerticalPole 的 ⟨*offset*⟩ 参数中，\Depth 将给出从基线 (baseline) 到相关 coffin 的边界框 (bounding box) 底部 (bottom) 的距离。

\Width \Width

在 \SetHorizontalPole 和 \SetVerticalPole 的 ⟨*offset*⟩ 参数中，\Width 将给出相关 coffin 的边界框 (bounding box) 从右边缘 (right edge) 到左边缘 (left edge) 的距离。

4 旋转 coffins

\RotateCoffin \RotateCoffin ⟨*coffin*⟩ {⟨*angle*⟩}

用给定的关于其参考点的 ⟨*angle*⟩ (逆时针方向给定的角度) 旋转 ⟨*coffin*⟩。这个过程将旋转 coffin 的内容 (content) 和杆 (poles)。多次旋转不会导致 coffin 的边界框 (bounding box) 不必要地增长 (growing)。

旋转对 coffin 的影响如图 2 所示。如图所示，coffin 的手柄 (handles) 将保持相对于 coffin 内容物的正确位置。旋转后 coffin 的“顶 (top)”当然可能不再是最接近物理页面顶部 (top of the physical page) 的边缘。

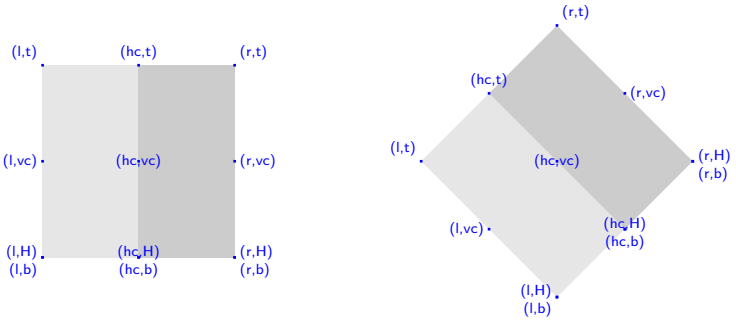

图 2: 旋转 Coffin：左图，未旋转；右图，已旋转 45°。

5 调整 coffins 的尺寸

\ResizeCoffin `\ResizeCoffin` ⟨*coffin*⟩ {⟨*width*⟩} {⟨*total-height*⟩}

将 ⟨*coffin*⟩ 重新调整 (resized) 到 ⟨*width*⟩ 和 ⟨*total-height*⟩，这两者都应作为尺寸表达式 (dimension expressions) 给出。

\ScaleCoffin `\ScaleCoffin` ⟨*coffin*⟩ {⟨*x-scale*⟩} {⟨*y-scale*⟩}

在水平方向 (horizontal directions) 和垂直方向 (vertical directions) 分别按 ⟨*x-scale*⟩ 和 ⟨*y-scale*⟩ 因子 (factors) 缩放 (scale) ⟨*coffin*⟩。这两个缩放因子 (scale factors) 应该作为实数 (real numbers) 给出。

`\ResizeCoffin` 和 `\ScaleCoffin` 可以互换使用：调整尺寸的最佳形式是比例因子 (scale factors) 还是绝对值 (absolute values) 将取决于上下文 (context)(图 3)。

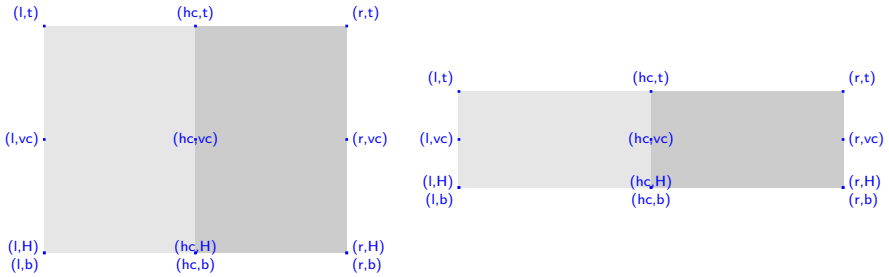

图 3: 调整 coffin 的尺寸：左图，调整尺寸为 4 cm 乘以 6 cm；右图，x 和 y 的比例因子分别为 2 和 0.5(如图 2 所示的 coffin 示例)。

6 连接 coffins

coffins 的关键操作 (key operation) 是将 coffins 相互连接。执行此操作时，第一个 coffin 始终是“父 coffin”，并通过对齐 (alignment) 进行更新。第二个“子 coffin”不会因对齐过程 (alignment process) 而改变。

\JoinCoffins `\JoinCoffins *`
⟨*coffin1*⟩ [⟨*coffin1-pole1*⟩ , ⟨*coffin1-pole2*⟩]
⟨*coffin2*⟩ [⟨*coffin2-pole1*⟩ , ⟨*coffin2-pole2*⟩]
(⟨*x-offset*⟩ , ⟨*y-offset*⟩)

两个 coffins 的连接由 `\JoinCoffins` 函数执行，该函数接受两个强制 (mandatory) 参数：“父”⟨*coffin1*⟩ 和“子”⟨*coffin2*⟩。显示的所有其他参数都是可选的 (optional)。

标准的 `\JoinCoffins` 函数将 ⟨*coffin2*⟩ 连接到 ⟨*coffin1*⟩，使得过程结束后的 ⟨*coffin1*⟩ 的边界框 (bounding box) 扩大。新的边界框将是覆盖两个输入 coffins 的边界框的最小矩形 (rectangle)。当使用 `\JoinCoffins` 的星号变体 (starred variant) 时，⟨*coffin1*⟩ 的边界框不会改变，即 ⟨*coffin2*⟩ 可能会突出于更新的 ⟨*coffin1*⟩ 的边界框之外。这两种对齐形式之间的区别最好用一个可视化示例 (visual example) 来说明。在图 4 中，对两个过程 (processes) 进行了对比。在这两种情况下，红色的小 coffin 都与灰色的大 coffin 对齐。在左边的插图中，使用了 `\JoinCoffins` 函数，从而产生了一个扩展的边界框 (expanded bounding box)。相比之下，右边的插图使用了 `\AttachCoffin`，这意味着边界框不包括较小 coffin 的区域。

图 4: `\JoinCoffins` (左图) 和 `\JoinCoffins*` (右图) 之间的对比；coffin 的边界框 (bounding box) 以黑色显示。

通过首先计算 ⟨*handle1*⟩，即 ⟨*coffin1-pole1*⟩ 和 ⟨*coffin1-pole2*⟩ 的交叉点，以及 ⟨*handle2*⟩，即 ⟨*coffin2-pole1*⟩ 和 ⟨*coffin2-pole2*⟩ 的交叉点，来执行对齐 (alignment)。如果这两个 ⟨*poles*⟩ 未指定，`\JoinCoffins` 将使用默认值 `(H,l)`，即 TEX 用于底层框 (underlying box) 的参考点 (reference point)。一旦找到了两个 ⟨*handles*⟩，⟨*coffin2*⟩ 就会附着到 ⟨*coffin1*⟩，使得 ⟨*handle1*⟩ 和 ⟨*handle2*⟩ 之间的关系由 ⟨*x-offset*⟩ 和 ⟨*y-offset*⟩ 来描述。这个 ⟨*offset*⟩ 是可选参数，如果未给出 ⟨*offset*⟩，则使用 `(0 pt, 0 pt)`。

注意，当使用 `\JoinCoffins` 时，新的边界框 (bounding box) 是包含两个输入 coffins 的边界框的最小矩形。因此，它将包括额外的空白 (additional white space)，除非一个 coffin 完全重叠其他 (图 5，左图)。在重新计算边界框时，coffins 的旋转将考虑旋转后材料的范围 (extent)。这意味着在旋转时不会添加 不必要 (*unnecessary*) 的空白 (图 5，右图)。

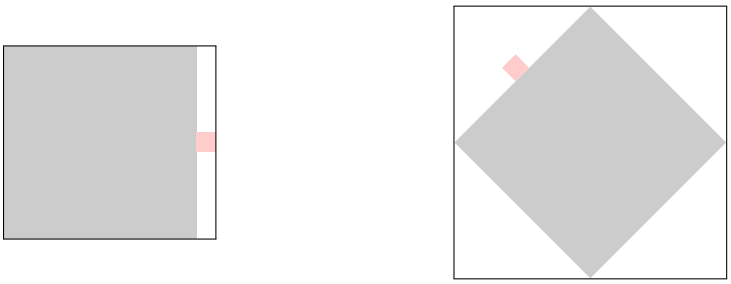

图 5: 已连接的 coffin 的旋转效果：黑线显示 coffin 的边界框。

作为连接过程 (joining procedure) 的一部分，两个输入 coffins 的杆 (poles) 被保存在更新 coffin 的结构 (structure) 内。这样就有可能执行复杂的对齐程序 (alignment procedures)。因此，对齐后的 coffin 的杆可分为三组 (groups)：

1. 更新 coffin 的“原始 (native)”杆，如 `l`、`r`、`hc` 等。
2. 源自 ⟨*coffin1*⟩ 的杆，如 ⟨*coffin1*⟩`-l`、⟨*coffin1*⟩`-r`、⟨*coffin1*⟩`-hc` 等。
3. 源自 ⟨*coffin2*⟩ 的杆，如 ⟨*coffin2*⟩`-l`、⟨*coffin2*⟩`-r`、⟨*coffin2*⟩`-hc` 等。

应用此功能可以进行一系列连接操作 (joining operations)，如图 6 所示。在本例中，用于对齐的方案 (scheme) 如下：

```
\SetHorizontalCoffin\OutputCoffin{}
\SetHorizontalCoffin\RedCoffin
  {\color{red!20!white}\rule{0.2 in}{0.2 in}}
\JoinCoffins\OutputCoffin[vc,hc]\RedCoffin[vc,hc]
\SetHorizontalCoffin\BlueCoffin
  {\color{blue!20!white}\rule{0.2 in}{0.2 in}}
\JoinCoffins\OutputCoffin[\RedCoffin-vc,\RedCoffin-hc]
  \BlueCoffin[b,l]
\SetHorizontalCoffin\GreenCoffin
  {\color{green!20!white}\rule{0.2 in}{0.2 in}}
\JoinCoffins\OutputCoffin[\BlueCoffin-vc,\BlueCoffin-hc]
  \GreenCoffin[b,l]
\SetHorizontalCoffin\YellowCoffin
  {\color{yellow!20!white}\rule{0.2 in}{0.2 in}}
\JoinCoffins\OutputCoffin[\GreenCoffin-vc,\GreenCoffin-hc]
  \YellowCoffin[b,l]
```

```
\SetHorizontalCoffin \OrangeCoffin
  {\color{orange!20!white}\rule{0.2 in}{0.2 in}}
\JoinCoffins\OutputCoffin[\BlueCoffin-t,\BlueCoffin-l]
  \OrangeCoffin[b,r]
\TypesetCoffin\OutputCoffin
```

此过程首先设置 `\OutputCoffin` 以保存连接的输出 (joined output)。然后，每个连接都会发生相对于前一个连接放置新添加项 (new addition) 的过程。由于每个连接的 coffin 都有一个唯一的名称，因此可以相对于组件 (assembly) 的每一个组成部分 (component parts) 进行对齐。这可以通过基于 `\BlueCoffin` 的早期位置 (earlier placement) 添加最终的 `\OrangeCoffin` 来说明。

图 6: 使用以前操作中的杆 (poles) 对齐 coffins。

7 排版 coffins

`\TypesetCoffin` `\TypesetCoffin`
  ⟨*coffin*⟩ `[` ⟨*pole1*⟩ `,` ⟨*pole2*⟩ `]`
  `(` ⟨*x-offset*⟩ `,` ⟨*y-offset*⟩ `)`

通过首先计算 ⟨*handle*⟩，即 ⟨*pole1*⟩ 和 ⟨*pole2*⟩ 的交叉点来排版。这是一个可选参数，如果未给出，则使用 `(H,l)`，即底层框 (underlying box) 的 TEX 参考点 (reference point)。然后对 coffin 进行排版，以使文档中的当前参考点与 ⟨*handle*⟩ 由 ⟨*x-offset*⟩ 和 ⟨*y-offset*⟩ 描述。这 ⟨*offset*⟩ 是可选的，如果未给出 ⟨*offset*⟩，则使用 `(0pt, 0pt)`。因此，排版 coffin 类似于执行一个对齐 (alignment)，其中“父 (parent)”coffin 是当前的插入点 (insertion point)。

8 测量 coffins

在设计过程 (design process) 中，有些地方可以在杆设置程序 (pole-setting procedures) 之外测量 coffins。

`\CoffinDepth` `\CoffinDepth` ⟨*coffin*⟩

以适合在 ⟨*dimension expression*⟩ 中使用的形式计算 ⟨*coffin*⟩ 的深度 (depth，低于基线)，例如，`\setlength{\mylength}{\CoffinDepth\ExampleCoffin}`。

\CoffinHeight `\CoffinHeight` ⟨*coffin*⟩

以适合在 ⟨*dimension expression*⟩ 中使用的形式计算 ⟨*coffin*⟩ 的高度 (height，高于基线)，例如，`\setlength{\mylength}{\CoffinHeight\ExampleCoffin}`。

\CoffinTotalHeight `\CoffinTotalHeight` ⟨*coffin*⟩

以适合在 ⟨*dimension expression*⟩ 中使用的形式计算 ⟨*coffin*⟩ 的总高度 (total height)，例如，`\setlength{\mylength}{\CoffinTotalHeight\ExampleCoffin}`。

\CoffinWidth `\CoffinWidth` ⟨*coffin*⟩

以适合在 ⟨*dimension expression*⟩ 中使用的形式计算 ⟨*coffin*⟩ 的宽度 (width)，例如，`\setlength{\mylength}{\CoffinWidth\ExampleCoffin}`。

9 诊断函数

用于跟踪 (following) coffin 建造过程 (coffin-building process) 的诊断数据 (diagnostic data) 可通过图形 (graphically) 和终端 (terminal) 获得。这反映了 coffins 是视觉构造 (visual constructs) 的事实。

\DisplayCoffinHandles `\DisplayCoffinHandles` ⟨*coffin*⟩ `{`⟨*color*⟩`}`

这个函数首先计算 ⟨*coffin*⟩ 的所有 ⟨*poles*⟩ 之间的交点，给出一组 ⟨*handles*⟩。然后，在源的当前位置 (current location in the source) 打印 ⟨*coffin*⟩，并将 ⟨*handles*⟩ 的位置 (position) 标记在 coffin 上。⟨*handles*⟩ 将被标记为这个过程 (process) 的一部分：⟨*handles*⟩ 的位置 (locations) 和标签 (labels) 都以指定的 ⟨*color*⟩ 被打印。

\MarkCoffinHandle `\MarkCoffinHandle` ⟨*coffin*⟩
`[` ⟨*pole1*⟩ `,` ⟨*pole2*⟩ `]` `{`⟨*color*⟩`}`

这个函数首先计算由 ⟨*pole1*⟩ 和 ⟨*pole2*⟩ 的交点所定义的 ⟨*coffin*⟩ 的 ⟨*handle*⟩。然后标记出 ⟨*handle*⟩ 在 ⟨*coffin*⟩ 上的位置。⟨*handle*⟩ 将被标记为此过程的一部分：⟨*handle*⟩ 的位置 (location) 和标签 (label) 都以指定的 ⟨*color*⟩ 被打印。如果没有给出 ⟨*poles*⟩，则使用缺省值 `(H,l)`。

\ShowCoffinStructure `\ShowCoffinStructure` ⟨*coffin*⟩

这个函数在终端 (terminal) 显示了 ⟨*coffin*⟩ 的结构信息 (structural information)。给出了排版材料 (typeset material) 的宽度 (width)、高度 (height) 和深度 (depth)，以及 coffin 所有杆 (location) 的位置。例如，对于图 2 中的已旋转的 (rotated) coffin，`\ShowCoffinStructure` 的输出是：

```
Size of coffin \ExampleCoffin:
> ht = 72.26999pt
> dp = 0.0pt
> wd = 72.26999pt
Poles of coffin \ExampleCoffin:
>  l  =>  {0pt}{0pt}{0pt}{1000pt}
>  B  =>  {0pt}{0pt}{1000pt}{0pt}
>  H  =>  {0pt}{0pt}{1000pt}{0pt}
>  T  =>  {0pt}{0pt}{1000pt}{0pt}
>  hc  =>  {36.135pt}{0pt}{0pt}{1000pt}
>  r  =>  {72.26999pt}{0pt}{0pt}{1000pt}
>  vc  =>  {0pt}{36.135pt}{1000pt}{0pt}
>  t  =>  {0pt}{72.26999pt}{1000pt}{0pt}
>  b  =>  {0pt}{0.0pt}{1000pt}{0pt}.
<recently read> }
```

请注意 coffin 的杆 (poles) 由四个值定义：杆通过的点的 x 和 y 坐标 (co-ordinates) 以及表示杆方向的向量 (vector) 的 x-分量和 y-分量。决定杆方向的是后者之间的比值 (ratio)，而不是绝对值 (absolute values)。

10 代码实现

```
1 ⟨*package⟩
2 ⟨@@=coffin⟩
3 \ProvidesExplPackage{xcoffins}{2024-01-04}{}
4   {L3 Experimental design level coffins}
```

\l__coffin_A_hpole_tl
\l__coffin_A_vpole_tl
\l__coffin_B_hpole_tl
\l__coffin_B_vpole_tl
\l__coffin_bound_box_grow_bool
\l__coffin_hoffset_dim
\l__coffin_voffset_dim

对齐系统 (alignment system) 的键值定义 (Key–value definitions)。除了 `grow-bounding-box` 之外，所有这些都必须给定一个值。

```
5  \keys_define:nn { coffin }
6  {
7      coffin1-hpole     .tl_set:N         = \l__coffin_A_hpole_tl          ,
8      coffin1-hpole     .value_required:n = true                           ,
9      coffin1-vpole     .tl_set:N         = \l__coffin_A_vpole_tl          ,
10     coffin1-vpole     .value_required:n = true                           ,
11     coffin2-hpole     .tl_set:N         = \l__coffin_B_hpole_tl          ,
12     coffin2-hpole     .value_required:n = true                           ,
13     coffin2-vpole     .tl_set:N         = \l__coffin_B_vpole_tl          ,
14     coffin2-vpole     .value_required:n = true                           ,
15     grow-bounding-box .bool_set:N       = \l__coffin_bound_box_grow_bool ,
16     grow-bounding-box .default:n        = true                           ,
17     hoffset           .dim_set:N        = \l__coffin_hoffset_dim         ,
18     hoffset           .value_required:n = true                           ,
19     voffset           .dim_set:N        = \l__coffin_voffset_dim         ,
20     voffset           .value_required:n = true
21 }
22 \keys_set:nn { coffin }
23 {
24     coffin1-hpole     = H    ,
25     coffin1-vpole     = l    ,
26     coffin2-hpole     = H    ,
27     coffin2-vpole     = l    ,
28     grow-bounding-box = true ,
29     hoffset           = 0 pt ,
30     voffset           = 0 pt
31 }
```

(*\l__coffin_A_hpole_tl 以及其它的定义结束。*)

__coffin_design_names:N
\Height
\Depth
\Width
\TotalHeight
\l__coffin_height_dim
\l__coffin_depth_dim
\l__coffin_width_dim
\l__coffin_totalheight_dim

为各种 coffin 尺寸 (dimensions) 设置设计级别的名称 (design-level names)。它们不是在这个范围 (scope) 之外定义的，而是尺寸，以便它们在例如 `\fp_eval:n` 这样的内部能正确工作 (work correctly)。

```
\cs_new_protected:Npn \__coffin_design_names:N #1
{
    \dim_set:Nn \l__coffin_height_dim { \coffin_ht:N #1 }
    \dim_set:Nn \l__coffin_depth_dim  { \coffin_dp:N #1 }
    \dim_set:Nn \l__coffin_width_dim  { \coffin_wd:N #1 }
    \dim_set:Nn \l__coffin_totalheight_dim
    { \l__coffin_height_dim + \l__coffin_depth_dim }
    \cs_set_eq:NN \Height \l__coffin_height_dim
    \cs_set_eq:NN \Depth \l__coffin_depth_dim
    \cs_set_eq:NN \Width \l__coffin_width_dim
    \cs_set_eq:NN \TotalHeight \l__coffin_totalheight_dim
}
\dim_new:N \l__coffin_height_dim
\dim_new:N \l__coffin_depth_dim
\dim_new:N \l__coffin_width_dim
\dim_new:N \l__coffin_totalheight_dim
```

(*__coffin_design_names:N* 以及其它的定义结束。这些函数被记录在第4页。)

其中大部分或多或少只是直接传递数据 (passing data straight through)。

\NewCoffin 这是一个非常简单的转换 (conversion)。

```
\NewDocumentCommand \NewCoffin { m }
{ \coffin_new:N #1 }
```

(*\NewCoffin* 定义结束。这个函数被记录在第3页。)

\SetHorizontalCoffin
\SetVerticalCoffin 这些都是直接的翻译 (straight-forward translations)。

```
\NewDocumentCommand \SetHorizontalCoffin { m +m }
{ \hcoffin_set:Nn #1 {#2} }
\NewDocumentCommand \SetVerticalCoffin { m m +m }
{ \vcoffin_set:Nnn #1 {#2} {#3} }
```

(*\SetHorizontalCoffin* 和 *\SetVerticalCoffin* 定义结束。这些函数被记录在第3页。)

\SetHorizontalPole
\SetVerticalPole 这里，需要为 coffin 尺寸设置设计级别名称 (design-level names)。这需要分组 (grouping)，但 coffin 工作 (work) 必须在组之外 (outside of the group) 进行。因此，这里有一点展开技巧 (expansion trickery)。

```
\NewDocumentCommand \SetHorizontalPole { m m m }
  {
    \group_begin:
      \__coffin_design_names:N #1
    \use:e
      {
```

```
        \group_end:
        \coffin_set_horizontal_pole:Nnn #1
          { \exp_not:n {#2} } { \dim_eval:n {#3} }
      }
  }
\NewDocumentCommand \SetVerticalPole { m m m }
  {
    \group_begin:
      \__coffin_design_names:N #1
    \use:e
      {
        \group_end:
        \coffin_set_vertical_pole:Nnn #1
          { \exp_not:n {#2} } { \dim_eval:n {#3} }
      }
  }
```

(*\SetHorizontalPole* 和 *\SetVerticalPole* 定义结束。这些函数被记录在第4页。)

\JoinCoffins

\JoinCoffins 函数需要对输入语法 (input syntax) 做一些工作，因为有许多可选参数 (optional arguments) 需要考虑。这里的想法是 **\JoinCoffins** 既可以用来扩展 ⟨*coffin1*⟩ 的边界框，也可以在不扩展边界框的情况下添加 ⟨*coffin2*⟩。还有两个手柄位置 (handle positions) 和偏移量 (offset) 要排序。

```
\NewDocumentCommand \JoinCoffins
  {
    o
    s
    m
    > { \SplitArgument { 1 } { , } } O { H , 1 }
    m
    > { \SplitArgument { 1 } { , } } O { H , 1 }
    > { \SplitArgument { 1 } { , } } D ( ) { 0 pt , 0 pt }
  }
  {
    \IfNoValueTF {#1}
      {
        \IfBooleanTF #2
          { \coffin_attach:NnnNnnnn #3 #4 #5 #6 #7 }
          { \coffin_join:NnnNnnnn #3 #4 #5 #6 #7 }
      }
      {
        \group_begin:
```

```
          \keys_set:nn { coffin } {#1}
          \tl_set:Ne \l__coffin_tmp_tl
            {
              \group_end:
              \bool_if:NTF \l__coffin_bound_box_grow_bool
                { \coffin_join:NnnNnnnn }
                { \coffin_attach:NnnNnnnn }
              \exp_not:N #3
              { \exp_not:o { \l__coffin_A_hpole_tl } }
              { \exp_not:o { \l__coffin_A_vpole_tl } }
              \exp_not:N #5
              { \exp_not:o { \l__coffin_B_hpole_tl } }
              { \exp_not:o { \l__coffin_B_vpole_tl } }
              { \dim_use:N \l__coffin_hoffset_dim }
              { \dim_use:N \l__coffin_voffset_dim }
            }
        \l__coffin_tmp_tl
      }
  }
```

(*\JoinCoffins* 定义结束。这个函数被记录在第6页。)

\TypesetCoffin 对于 coffins 的排版，有两个可选参数，这两个参数都需要拆分 (split)。这是 \JoinCoffins 所需代码的一个更简单的例子。

```
\NewDocumentCommand \TypesetCoffin
{
    m
    > { \SplitArgument { 1 } { , } } O { H , 1 }
    > { \SplitArgument { 1 } { , } } D ( ) { 0 pt , 0 pt }
}
{ \coffin_typeset:Nnnnn #1 #2 #3 }
```

(*\TypesetCoffin* 定义结束。这个函数被记录在第9页。)

\RotateCoffin 更多直接拷贝 (straight-forward copies)。

\ResizeCoffin

\ScaleCoffin

```
\NewDocumentCommand \RotateCoffin  { m m }
{ \coffin_rotate:Nn #1 {#2} }
\NewDocumentCommand \ResizeCoffin  { m m m }
{ \coffin_resize:Nnn #1 {#2} {#3} }
\NewDocumentCommand \ScaleCoffin  { m m m }
{ \coffin_scale:Nnn #1 {#2} {#3} }
```

(*\RotateCoffin*, *\ResizeCoffin*, 和 *\ScaleCoffin* 定义结束。这些函数被记录在第5页。)

\CoffinDepth
\CoffinHeight
\CoffinTotalHeigth
\CoffinWidth

没有什么太复杂的，除了将总高度 (total height) 设置为表达式 (expression)，以便在前缀 (prefixed) 为负号 (negative sign) 等时能正确操作。

```
\NewDocumentCommand \CoffinDepth { m }
{ \dim_eval:n { \coffin_dp:N #1 } }
\NewDocumentCommand \CoffinHeight { m }
{ \dim_eval:n { \coffin_ht:N #1 } }
\NewDocumentCommand \CoffinTotalHeight { m }
{ \dim_eval:n { \coffin_ht:N #1 + \coffin_dp:N #1 } }
\NewDocumentCommand \CoffinWidth { m }
{ \dim_eval:n { \coffin_wd:N #1 } }
```

(*\CoffinDepth* 以及其它的定义结束。这些函数被记录在第9页。)

\DisplayCoffinHandles

显示所有手柄 (handles) 稍微容易一点，因为不必担心手柄。

```
\NewDocumentCommand \DisplayCoffinHandles { m m }
{ \coffin_if_exist:NT #1 { \coffin_display_handles:Nn #1 {#2} } }
```

(*\DisplayCoffinHandles* 定义结束。这个函数被记录在第10页。)

\MarkCoffinHandle

标记手柄 (marking a handle) 需要对输入 (input) 进行一些处理，这样设计级别的接口 (design-level interface) 就“漂亮 (nice)”了。

```
\NewDocumentCommand \MarkCoffinHandle
{ m > { \SplitArgument { 1 } { , } } O { H , l } m }
{  \coffin_if_exist:NT #1 { \coffin_mark_handle:Nnnn #1 #2 {#3} } }
```

(*\MarkCoffinHandle* 定义结束。这个函数被记录在第10页。)

\ShowCoffinStructure

再次回到易于实现的函数 (easy-to-implement functions)。

```
\NewDocumentCommand \ShowCoffinStructure { m }
{ \coffin_show_structure:N #1 }
```

(*\ShowCoffinStructure* 定义结束。这个函数被记录在第11页。)

⟨/package⟩

索引

斜体数字指向相应条目描述的页面，下划线数字指向定义的代码行，其它的都指向使用条目的页面。