第五章 Boxes
盒子

The horizontal and vertical boxes of TEX are containers for pieces of horizontal and vertical lists. Boxes can be stored in box registers. This chapter treats box registers and such aspects of boxes as their dimensions, and the way their components are placed relative to each other.

TEX 的水平盒子和垂直盒子是水平和垂直列表的容器。盒子可以存储在盒子寄存器中。本章介绍盒子寄存器以及盒子的尺寸和它们的组成部分相对于彼此的放置方式等方面。

\hbox Construct a horizontal box.

构建一个水平盒子。

\vbox Construct a vertical box with reference point of the last item.

构建一个垂直盒子，其参考点位于最后一个项目。

\vtop Construct a vertical box with reference point of the first item.

构建一个垂直盒子，其参考点位于第一个项目。

\vcenter Construct a vertical box vertically centred on the math axis; this command can only be used in math mode.

构建一个在数学轴上垂直居中的垂直盒子；此命令只能在数学模式中使用。

\vsplit Split off the top part of a vertical box.

将垂直盒子的顶部部分分离出来。

\box Use a box register, emptying it.

使用一个盒子寄存器，并清空其内容。

\setbox Assign a box to a box register.

将盒子赋值给一个盒子寄存器。

\copy Use a box register, but retain the contents.
使用一个盒子寄存器，但保留其内容。

\ifhbox \ifvbox Test whether a box register contains a horizontal/vertical box.
检查一个盒子寄存器是否包含一个水平/垂直盒子。

\ifvoid Test whether a box register is empty.
检查一个盒子寄存器是否为空。

\newbox Allocate a new box register.
分配一个新的盒子寄存器。

\unhbox \unvbox Unpack a box register containing a horizontal/vertical box, adding the contents to the current horizontal/vertical list, and emptying the register.
解开包含水平/垂直盒子的盒子寄存器，将其内容添加到当前的水平/垂直列表中，并清空寄存器。

\unhcopy \unvcopy The same as \unhbox / \unvbox, but do not empty the register.
与 \unhbox,/,\unvbox 相同，但不清空寄存器。

\ht \dp \wd Height/depth/width of the box in a box register.
盒子寄存器中盒子的高度/深度/宽度。

\boxmaxdepth Maximum allowed depth of boxes. Plain TEX default: \maxdimen.
盒子的最大允许深度。Plain TEX 的默认值：\maxdimen。

\splitmaxdepth Maximum allowed depth of boxes generated by \vsplit.
\vsplit 生成的盒子的最大允许深度。

\badness Badness of the most recently constructed box.
最近构建的盒子的劣度。

\hfuzz \vfuzz Excess size that TEX tolerates before it considers
在 TEX 认为一个水平/垂直盒子过满之前，允许的多余尺寸。a horizontal/ vertical box overfull.

\hbadness \vbadness Amount of tolerance before TEX reports an underfull
在 TEX 报告一个水平/垂直盒子不充分或过充分之前的容忍度。or overfull horizontal/vertical box.

\overfullrule Width of the rule that is printed to indicate overfull horizontal boxes.

用于指示过度充满水平盒子的标尺的宽度。

\hsize Line width used for text typesetting inside a vertical box.

用于垂直盒子内文本排版的行宽。

\vsize Height of the page box.

页面盒子的高度。

\lastbox Register containing the last item added to the current list, if this was a box.

寄存器，其中包含添加到当前列表中的最后一个项目，如果该项目是一个盒子。

\raise \lower Adjust vertical positioning of a box in horizontal mode.

调整水平模式下盒子的垂直位置。

\moveleft \moveright Adjust horizontal positioning of a box in vertical mode.

调整垂直模式下盒子的水平位置。

\everyhbox \everyvbox Token list inserted at the start of a horizontal/vertical box.

插入到水平/垂直盒子开始处的记号列表。

5.1 Boxes
盒子

In this chapter we shall look at boxes. Boxes are containers for pieces of horizontal or vertical lists. Boxes that are needed more than once can be stored in box registers.

在本章中，我们将讨论盒子（boxes）。盒子是水平或垂直列表的容器。需要多次使用的盒子可以存储在盒子寄存器中。

When TEX expects a ⟨box⟩, any of the following forms is admissible:

当 TEX 需要一个 ⟨box⟩ 时，可以使用以下任意形式：

- \hbox⟨box specification⟩{⟨horizontal material⟩}
- \vbox⟨box specification⟩{⟨vertical material⟩}
- \vtop⟨box specification⟩{⟨vertical material⟩}

- \box⟨8-bit number⟩
- \copy⟨8-bit number⟩
- \vsplit⟨8-bit number⟩to⟨dimen⟩
- \lastbox

A ⟨box specification⟩ is defined as

⟨box specification⟩ 的定义为:

⟨box specification⟩ ⟶ ⟨filler⟩
| to ⟨dimen⟩⟨filler⟩ | spread ⟨dimen⟩⟨filler⟩

An ⟨8-bit number⟩ is a number in the range 0–255.

⟨8-bit number⟩ 是范围在 0–255 的数字。

The braces surrounding box material define a group; they can be explicit characters of categories 1 and 2 respectively, or control sequences \let to such characters; see also below.

括号将盒子内容定义为一组；它们可以是类别码为 1 和 2 的显式字符，也可以是被 \let 给予这些字符的控制序列；详见下文。

A ⟨box⟩ can in general be used in horizontal, vertical, and math mode, but see below for the \lastbox. The connection between boxes and modes is explored further in Chapter ??.

⟨box⟩ 通常可以在水平、垂直和数学模式中使用，但请参见下文有关 \lastbox 的内容。盒子与模式之间的关系将在第 ?? 章中进一步探讨。

The box produced by \vcenter – a command that is allowed only in math mode – is not a ⟨box⟩. For instance, it can not be assigned with \setbox; see further Chapter ??.

由 \vcenter 生成的盒子（该命令仅在数学模式下允许）不是 ⟨box⟩。例如，它不能使用 \setbox 进行赋值；详见第 ?? 章。

The \vsplit operation is treated in Chapter ??.

\vsplit 操作将在第 ?? 章中介绍。

5.2　Box registers

盒子寄存器

There are 256 box registers, numbered 0–255. Either a box register is empty (‘void’), or it contains a horizontal or vertical box. This section discusses specifically box registers; the sizes of boxes, and the way material is arranged inside them, is treated below.

有 256 个盒子寄存器，编号为 0–255。一个盒子寄存器可以是空的（“void”），也可以包含一个水平盒子或垂直盒子。本节具体讨论盒子寄存器；盒子的尺寸和其中的内容排列方式将在下面讨论。

5.2.1 Allocation: \newbox
分配：\newbox

The plain TEX \newbox macro allocates an unused box register:

在 plain TEX 中，\newbox 宏用于分配一个未使用的盒子寄存器：

\newbox\MyBox

after which one can say

然后可以这样使用：

\setbox\MyBox=...

or 或者

\box\MyBox

and so on. Subsequent calls to this macro give subsequent box numbers; this way macro collections can allocate their own boxes without fear of collision with other macros.

等等。后续对此宏的调用会给出连续的盒子编号；这样，宏集合可以分配自己的盒子，而不用担心与其他宏发生冲突。

The number of the box is assigned by \chardef (see Chapter ??). This implies that \MyBox is equivalent to, and can be used as, a ⟨number⟩. The control sequence \newbox is an \outer macro. Newly allocated box registers are initially empty.

盒子的编号由 \chardef 分配（见第 ?? 章）。这意味着 \MyBox 等同于并且可以用作一个 ⟨number⟩。\newbox 控制序列是一个 \outer 宏。新分配的盒子寄存器最初是空的。

5.2.2　Usage: \setbox, \box, \copy 用法：\setbox、\box、\copy

A register is filled by assigning a ⟨box⟩ to it:

通过将一个 ⟨box⟩ 赋值给寄存器来填充它：

\setbox⟨number⟩⟨equals⟩⟨box⟩

For example, the ⟨box⟩ can be explicit

例如，⟨box⟩ 可以是显式的

\setbox37=\hbox{...}　or　\setbox37=\vbox{...}

or it can be a box register:

例如，⟨box⟩ 可以是显式的

\setbox37=\box38

Usually, box numbers will have been assigned by a \newbox command.

通常，盒子编号将通过 \newbox 命令进行分配。

The box in a box register is appended by the commands \box and \copy to whatever list TEX is building: the call

通过 \box 和 \copy 命令，盒子寄存器中的盒子被附加到 TEX 正在构建的任何列表中：调用

\box38

appends box 38. To save memory space, box registers become empty by using them: TEX assumes that after you have inserted a box by calling \box*nn* in some mode, you do not need the contents of that register any more and empties it. In case you do need the contents of a box register more than once, you can \copy it. Calling \copy*nn* is equivalent to \box*nn* in all respects except that the register is not cleared.

将附加盒子 38。为了节省内存空间，通过使用盒子寄存器来使其为空：TEX 假定在你通过在某种模式下调用 \box*nn* 插入盒子后，你不再需要该寄存器的内容，并将其清空。如果您需要多次使用盒子寄存器的内容，可以使用 \copy 来复制它。调用 \copy*nn* 在所有方面等效于 \box*nn*，只是寄存器不会被清空。

It is possible to unwrap the contents of a box register by 'unboxing' it using the commands \unhbox and \unvbox, and their copying versions \unhcopy and \unvcopy. Whereas a box can be used in any mode, the unboxing operations can only be used in the appropriate mode, since in effect they contribute a partial horizontal or vertical list (see also Chapter ??). See below for more information on unboxing registers.

可以通过使用 \unhbox 和 \unvbox 命令及其拷贝版本 \unhcopy 和 \unvcopy 来"解包"盒子寄存器的内容。虽然盒子可以在任何模式下使用，但解包操作只能在适当的模式下使用，因为实际上它们贡献了一个部分的水平或垂直列表（也请参见第 ?? 章）。有关在解包寄存器上的更多信息，请参见下文。

5.2.3 Testing: \ifvoid, \ifhbox, \ifvbox 测试：\ifvoid、\ifhbox、\ifvbox

Box registers can be tested for their contents:

可以测试盒子寄存器的内容：

\ifvoid⟨number⟩

is true if the box register is empty. Note that an empty, or 'void', box register is not the same as a register containing an empty box. An empty box is still either a horizontal or a vertical box; a void register can be used as both.

如果盒子寄存器为空，则为真。请注意，空的或"空"的盒子寄存器不同于包含空盒子的寄存器。空盒子仍然是水平或垂直盒子；空的寄存器可以用作两者。

The test 测试

\ifhbox⟨number⟩

is true if the box register contains a horizontal box;

如果盒子寄存器包含水平盒子，则为真；

\ifvbox⟨number⟩

is true if the box register contains a vertical box. Both tests are false for void registers.

如果盒子寄存器包含垂直盒子，则为真。对于空寄存器，这两个测试都为假。

5.2.4　The \lastbox

When TEX has built a partial list, the last box in this list is accessible as the \lastbox. This behaves like a box register, so you can remove the last box from the list by assigning the \lastbox to some box register. If the last item on the current list is not a box, the \lastbox acts like a void box register. It is not possible to get hold of the last box in the case of the main vertical list. The \lastbox is then always void.

当 TEX 构建了一个部分列表时，该列表中的最后一个盒子可以通过 \lastbox 访问。它的行为类似于一个盒子寄存器，因此可以通过将 \lastbox 赋值给某个盒子寄存器来从列表中删除最后一个盒子。如果当前列表上的最后一个项目不是一个盒子，则 \lastbox 的行为类似于一个空的盒子寄存器。在主垂直列表的情况下，无法获取到最后一个盒子。此时，\lastbox 总是空的。

As an example, the statement

举个例子，语句

{\setbox0=\lastbox}

removes the last box from the current list, assigning it to box register 0. Since this assignment occurs inside a group, the register is cleared at the end of the group. At the start of a paragraph this can be used to remove the indentation box (see Chapter ??). Another example of \lastbox can be found on page 1.

从当前列表中删除最后一个盒子，并将其赋值给盒子寄存器 0。由于此赋值发生在组内，因此该寄存器在组结束时会被清空。在段落的开头，可以使用它来删除缩进盒子（参见第 ??章）。另一个关于 \lastbox 的例子可以在第1 页找到。

Because the \lastbox is always empty in external vertical mode, it is not possible to get hold of boxes that have been added to the page. However, it is possible to dissect the page once it is in \box255, for instance doing

由于在外部垂直模式中 \lastbox 总是空的，所以无法获取已添加到页面上的盒子。但是，在输出例程中可以对页面进行分析，例如在输出例程内部执行以下操作：

```
\vbox{\unvbox255{\setbox0=\lastbox}}
```

inside the output routine.

If boxes in vertical mode have been shifted by \moveright or \moveleft, or if boxes in horizontal mode have been raised by \raise or lowered by \lower, any information about this displacement due to such a command is lost when the \lastbox is taken from the list.

如果垂直模式中的盒子通过 \moveright 或 \moveleft 进行了偏移，或者水平模式中的盒子通过 \raise 进行了升高或 \lower 进行了降低，那么当从列表中获取 \lastbox 时，关于这些命令引起的偏移的任何信息都会丢失。

5.3 Natural dimensions of boxes 盒子的自然尺寸

5.3.1 Dimensions of created horizontal boxes 创建水平盒子的尺寸

Inside an \hbox all constituents are lined up next to each other, with their reference points on the baseline of the box, unless they are moved explicitly in the vertical direction by \lower or \raise.

在 \hbox 内部，所有组成部分都相互排列在一起，它们的参考点位于盒子的基线上，除非通过 \lower 或 \raise 明确在垂直方向上移动。

The resulting width of the box is the sum of the widths of the components. Thus the width of

盒子的宽度是组成部分的宽度之和。因此，

```
\hbox{\hskip1cm}
```

is positive, and the width of

的宽度是正的，而

\hbox{\hskip-1cm}

is negative. By way of example,

的宽度是负的。例如，

```
a\hbox{\kern-1em b}--
```

gives as output

的输出结果是

```
ba
```

which shows that a horizontal box can have negative width.

这表明水平盒子的宽度可以为负。

The height and depth of an \hbox are the maximum amount that constituent boxes project above and below the baseline of the box. They are non-negative when the box is created.

\hbox 的高度和深度是组成部分在基线之上和之下突出的最大量。当创建盒子时，它们是非负的。

The commands \lower and \raise are the only possibilities for vertical movement inside an \hbox (other than including a \vbox inside the \hbox, of course); a ⟨vertical command⟩ – such as \vskip – is not allowed in a horizontal box, and \par, although allowed, does not do anything inside a horizontal box.

\lower 和 \raise 命令是在 \hbox 内部进行垂直移动的唯一可能性（当然，除了在 \hbox 中包含 \vbox）；⟨vertical command⟩（如 \vskip）不允许在水平盒子中，而且虽然允许使用 \par，但在水平盒子中它不起作用。

5.3.2　Dimensions of created vertical boxes 创建垂直盒子的尺寸

Inside a \vbox vertical material is lined up with the reference points on the vertical line through the reference point of the box, unless components are moved explicitly in the horizontal direction by \moveleft or \moveright.

在 \vbox 内部，垂直材料与盒子参考点通过盒子的参考点的垂直线对齐，除非组成部分通过 \moveleft 或 \moveright 在水平方向上明确移动。

The reference point of a vertical box is always located at the left boundary of the box. The width of a vertical box is then the maximal amount that any material in the box sticks to the right of the reference point. Material to the left of the reference point is not taken into account in the width. Thus the result of

垂直盒子的参考点总是位于盒子的左边界处。垂直盒子的宽度是盒子中任何内容向右粘附的最大量。参考点左侧的内容不计入宽度。因此，

```
a\vbox{\hbox{\kern-1em b}}--
```

is 的结果是

ba–

This should be contrasted with the above example.

这与上面的示例形成对比。

The calculation of height and depth is different for vertical boxes constructed by \vbox and \vtop. The ground rule is that a \vbox has a reference point that lies on the baseline of its last component, and a \vtop has its reference point on the baseline of the first component. In general, the depth (height) of a \vbox (\vtop) can be non-zero if the last (first) item is a box or rule.

对于由 \vbox 和 \vtop 构造的垂直盒子，高度和深度的计算方式是不同的。基本规则是，\vbox 的参考点位于其最后一个组件的基线上，而 \vtop 的参考点位于第一个组件的基线上。一般来说，\vbox（\vtop）的深度（高度）可以是非零的，如果最后（第一个）的项目是一个盒子或标线的话。

The height of a \vbox is then the sum of the heights and depths of all components except the last, plus the height of that last component; the depth of the \vbox is the depth of its last component. The depth of a \vtop is the sum of the depth of the first component and the heights and depths of all subsequent material; its height is the height of the first component.

垂直盒子的高度是除最后一个组件外所有组件的高度和深度之和，加上最后一个组件的高度；垂直盒子的深度是其最后一个组件的深度。\vtop 的深度是第一个组件的深度与所有后续内容的高度和深度之和；其高度是第一个组件的高度。

However, the actual rules are a bit more complicated when the first component of a \vtop or the last component of a \vbox is not a box or rule. If the last component of a \vbox is a kern or a glue, the depth of that box is zero; a \vtop's height is zero unless its first component is a box or rule. (Note the asymmetry in these definitions; see below for an example illustrating this.) The depth of a \vtop, then, is equal to the total height plus depth of all enclosed material minus the height of the \vtop.

然而，当 \vtop 的第一个组件或 \vbox 的最后一个组件不是盒子或标线时，实际规则会更复杂一些。如果 \vbox 的最后一个组件是紧排或粘连，则该盒子的深度为零；\vtop 的高度为零，除非它的第一个组件是一个盒子或标线。（请注意这些定义中的不对称性；下面的示例将说明这一点。）因此，\vtop 的深度等于所有封闭内容的总高度加深度减去 \vtop 的高度。

There is a limit on the depth of vertical boxes: if the depth of a \vbox or \vtop calculated by the above rules would exceed , the reference point of the box is moved down by the excess amount. More precisely, the excess depth is added to the natural height of the box. If the box had a to or spread specification, any glue is set anew to take the new height into account.

垂直盒子有一个深度的限制：如果根据上述规则计算的 \vbox 或 \vtop 的深度超过 ，则盒子的基准点会向下移动超出的量。更确切地说，超出的深度会添加到盒子的自然高度上。如果盒子具有 to 或 spread 规范，则重新设置任何粘连以考虑新的高度。

Ordinarily, \boxmaxdepth is set to the maximum dimension possible in TEX. It is for instance reduced during some of the calculations in the plain TEX output routine; see Chapter ??.

通常情况下，\boxmaxdepth 被设置为 TEX 中可能的最大尺寸。例如，在 plain TEX 的输出例程的某些计算过程中可能会减小它；请参见第 ?? 章。

5.3.3 Examples 示例

Horizontal boxes are relatively straightforward. Their width is the distance between the 'beginning' and the 'end' of the box, and consequently the width is not necessarily positive. With

水平盒子相对简单。它们的宽度是盒子的“开始”和“结束”之间的距离，因此宽度不一定是正的。通过以下代码：

```
\setbox0=\hbox{aa} \setbox1=\hbox{\copy0 \hskip-\wd0}
```

the \box1 has width zero;

\box1 的宽度为零；

```
    /\box1/   gives   ‘/a̸a’
```

The height and depth of a horizontal box cannot be negative: in

水平盒子的高度和深度不能是负值：在以下代码中：

```
\setbox0=\hbox{\vrule height 5pt depth 5pt}
\setbox1=\hbox{\raise 10pt \box0}
```

the \box1 has depth 0pt and height 15pt

\box1 的深度为 0pt，高度为 15pt。

Vertical boxes are more troublesome than horizontal boxes. Let us first treat their width. After

垂直盒子比水平盒子更棘手。我们首先来讨论它们的宽度。在执行

```
\setbox0=\hbox{\hskip 10pt}
```

the box in the \box0 register has a width of 10pt. Defining

之后，盒子寄存器 \box0 中的盒子宽度为 10pt。定义

```
\setbox1=\vbox{\moveleft 5pt \copy0}
```

the \box1 will have width 5pt; material to the left of the reference point is not accounted for in the width of a vertical box. With

则 \box1 的宽度为 5pt；参考点左侧的内容在垂直盒子的宽度中不计算在内。使用

```
\setbox2=\vbox{\moveright 5pt \copy0}
```

the \box2 will have width 15pt.

则 \box2 的宽度为 15pt。

The depth of a \vbox is the depth of the last item if that is a box, so

\vbox 的深度是最后一个项目的深度（如果该项目是一个盒子），因此

\vbox{\vskip 5pt \hbox{\vrule height 5pt depth 5pt}}

has height 10pt and depth 5pt, and

的高度为 10pt，深度为 5pt，而

\vbox{\vskip -5pt \hbox{\vrule height 5pt depth 5pt}}

has height 0pt and depth 5pt. With a glue or kern as the last item in the box, the resulting depth is zero, so

的高度为 0pt，深度为 5pt。如果在盒子中的最后一个项目是粘连或紧排，则结果的深度为零，因此

\vbox{\hbox{\vrule height 5pt depth 5pt}\vskip 5pt}

has height 15pt and depth 0pt;

的高度为 15pt，深度为 0pt；

\vbox{\hbox{\vrule height 5pt depth 5pt}\vskip -5pt}

has height 5pt and depth 0pt.

的高度为 5pt，深度为 0pt。

The height of a \vtop behaves (almost) the same with respect to the first item of the box, as the depth of a \vbox does with respect to the last item. Repeating the above examples with a \vtop gives the following:

\vtop 的高度与盒子的第一个项目的行为（几乎）相同，就像 \vbox 的深度与最后一个项目的行为一样。使用 \vtop 重复上面的示例得到以下结果：

\vtop{\vskip 5pt \hbox{\vrule height 5pt depth 5pt}}

has height 0pt and depth 15pt, and

的高度为 0pt，深度为 15pt，而

\vtop{\vskip -5pt \hbox{\vrule height 5pt depth 5pt}}

has height 0pt and depth 5pt;

的高度为 0pt，深度为 5pt；

\vtop{\hbox{\vrule height 5pt depth 5pt} \vskip 5pt}

has height 5pt and depth 10pt, and

的高度为 5pt，深度为 10pt，以及

\vtop{\hbox{\vrule height 5pt depth 5pt} \vskip -5pt}

has height 5pt and depth 0pt.

的高度为 5pt，深度为 0pt。

5.4 More about box dimensions 更多关于盒子尺寸的内容

5.4.1 Predetermined dimensions 预先确定的尺寸

The size of a box can be specified in advance with a ⟨box specification⟩; see above for the syntax. Any glue in the box is then set in order to reach the required size. Prescribing the size of the box is done by

可以预先通过 ⟨box specification⟩（参见上文的语法）指定盒子的尺寸。盒子中的任何粘连都将根据所需的尺寸进行设置。通过以下方式指定盒子的尺寸：

\hbox to ⟨dimen⟩ {...}, \vbox to ⟨dimen⟩ {...}

If stretchable or shrinkable glue is present in the box, it is stretched or shrunk in order to give the box the specified size. Associated with this glue setting is a badness value (see Chapter ??). If no stretch or shrink – whichever is necessary – is present, the resulting box will be underfull or overfull respectively. Error reporting for over/underfull boxes is treated below.

如果盒子中存在可伸缩或可收缩的粘连，则会拉伸或收缩粘连以使盒子达到指定的尺寸。与此粘连设置相关联的是一个劣度值（详见第 ?? 章）。如果没有可伸展或可收缩的部分（取决于需要的部分），则得到的盒子将分别是不充实或过充实的。下文将介绍关于过充实/不充实盒子的错误报告。

Another command to let a box have a size other than the natural size is

可以使用另一个命令来指定盒子具有与其自然尺寸不同的尺寸：

\hbox spread ⟨dimen⟩ {...}, \vbox spread ⟨dimen⟩ {...}

which tells TEX to set the glue in such a way that the size of the box is a specified amount more than the natural size.

这告诉 TEX 根据粘连设置方式设置粘连，使盒子的尺寸比其自然尺寸多出指定的量。

Box specifications for \vtop vertical boxes are somewhat difficult to interpret. TEX constructs a \vtop by first making a \vbox, including glue settings induced by a ⟨box specification⟩; then it computes the height and depth by the above rules. Glue setting is described in Chapter ??.

对于 \vtop 垂直盒子的盒子规格有些难以解释。TEX 通过首先创建一个 \vbox，包括由 ⟨box specification⟩ 引起的粘连设置，然后按照上述规则计算高度和深度来构造 \vtop。粘连的设置方法在第 ?? 章中有描述。

5.4.2 Changes to box dimensions 更改盒子尺寸

The dimensions of a box register are accessible by the commands \ht, \dp, and \wd; for instance \dp13 gives the depth of box 13. However, not only can boxes be measured this way; by assigning values to these dimensions TEX can even be fooled into thinking that a box has a size different from its actual. However, changing the dimensions of a box does not change anything about the contents; in particular it does not change the way the glue is set.

可以通过命令 \ht、\dp 和 \wd 访问盒子寄存器的尺寸；例如，\dp13 给出盒子 13 的深度。然而，不仅可以用这种方式测量盒子的尺寸；通过为这些尺寸分配值，甚至可以欺骗 TEX，使其认为盒子的尺寸与实际尺寸不同。然而，更改盒子的尺寸不会改变其内容；特别是，它不会改变粘连设置的方式。

Various formats use this in ‘smash’ macros: the macro defined by

各种格式在“smash”宏中使用了这一点：由下面定义的宏

```
\def\smash#1{{\setbox0=\hbox{#1}\dp0=0pt \ht0=0pt \box0\relax}}
```

places its argument but annihilates its height and depth; that is, the output does show the whole box, but further calculations by TEX act as if the height and depth were zero.

将其参数放置在盒子中，但会将其高度和深度消除；也就是说，输出显示整个盒子，但是 TEX 的进一步计算会将高度和深度视为零。

Box dimensions can be changed only by setting them. They are ⟨box dimen⟩s, which can only be set in a ⟨box size assignment⟩, and not, for instance changed with \advance.

只能通过设置它们来更改盒子寄存器的尺寸。它们是 ⟨box dimen⟩，只能在 ⟨box size assignment⟩ 中设置，不能通过 \advance 等方式更改。

Note that a ⟨box size assignment⟩ is a ⟨global assignment⟩: its effect transcends any groups in which it occurs (see Chapter ??). Thus the output of

请注意，⟨box size assignment⟩ 是一个 ⟨global assignment⟩：其影响超越了其所在的任何组（见第 ?? 章）。因此，以下代码的输出

\setbox0=\hbox{---} {\wd0=0pt} a\box0b

is 是 ‘ab–’.

The limits that hold on the dimensions with which a box can be created (see above) do not hold for explicit changes to the size of a box: the assignment \dp0=-2pt for a horizontal box is perfectly admissible.

适用于创建盒子的尺寸的限制（见上文）在显式更改盒子尺寸时不适用：例如，对于水平盒子，\dp0=””-2pt 是完全允许的。

5.4.3 Moving boxes around 移动盒子

In a horizontal box all constituent elements are lined up with their reference points at the same height as the reference point of the box. Any box inside a horizontal box can be lifted or dropped using the macros \raise and \lower.

在水平盒子中，所有组成元素都与盒子的基准点对齐。可以使用 \raise 和 \lower 宏将水平盒子内的任何盒子向上或向下移动。

Similarly, in a vertical box all constituent elements are lined up with their reference points underneath one another, in line with the reference point of the box. Boxes can now be moved sideways by the macros \moveleft and \moveright.

同样，在垂直盒子中，所有组成元素都与盒子的基准点在同一高度下方对齐。可以使用 \moveleft 和 \moveright 宏将盒子沿水平方向移动。

Only boxes can be shifted thus; these operations cannot be applied to, for instance, characters or rules.

只有盒子可以这样移动；不能将此操作应用于字符或规则等其他元素。

5.4.4 Box dimensions and box placement 盒子尺寸和盒子放置

TEX places the components of horizontal and vertical lists by maintaining a reference line and a current position on that line. For horizontal lists the reference line is the baseline of the surrounding \hbox; for vertical lists it is the vertical line through the reference point of the surrounding \vbox.

TEX 通过维护参考线和参考线上的当前位置来放置水平和垂直列表的组成部分。对于水平列表，参考线是周围 \hbox 的基线；对于垂直列表，它是通过周围 \vbox 的基准点的垂直线。

In horizontal mode a component is placed as follows. The current position coincides initially with the reference point of the surrounding box. After that, the following actions are carried out.

在水平模式下，组成部分的放置方式如下。当前位置最初与周围盒子的基准点重合。然后，执行以下操作。

1. If the component has been shifted by \raise or \lower, shift the current position correspondingly.
   如果组成部分已通过 \raise 或 \lower 进行了移动，则相应地移动当前位置。
2. If the component is a horizontal box, use this algorithm recursively for its contents; if it is a vertical box, go up by the height of this box, putting a new current position for the enclosed vertical list there, and place its components using the algorithm for vertical lists below.
   如果组成部分是水平盒子，请递归使用此算法处理其内容；如果是垂直盒子，则向上移动此盒子的高度，并在那里放置一个新的当前位置，然后使用垂直列表的算法放置其组成部分。
3. Move the current position (on the reference line) to the right by the width of the component.

将当前位置（在参考线上）向右移动组成部分的宽度。

For the list in a vertical box TEX's current position is initially at the upper left corner of that box, as explained above, and the reference line is the vertical line through that point; it also runs through the reference point of the box. Enclosed components are then placed as follows.

对于垂直盒子中的列表，TEX 的当前位置最初位于该盒子的左上角，如上所述，参考线是通过该点的垂直线；它也通过盒子的参考点。然后将封闭的组件放置如下：

1. If a component has been shifted using \moveleft or \moveright, shift the current position accordingly.
   如果组件已经使用 \moveleft 或 \moveright 进行了偏移，则相应地移动当前位置。
2. Put the component with its upper left corner at the current position.
   将组件的左上角放在当前位置。
3. If the component is a vertical box, use this algorithm recursively for its contents; if it is a horizontal box, its reference point can be found below the current position by the height of the box. Put the current position for that box there, and use the above algorithm for horizontal lists.
   如果组件是一个垂直盒子，则对其内容递归使用此算法；如果它是一个水平盒子，则通过该盒子的高度可以找到当前位置下方的参考点。将当前位置放在该盒子的参考点处，并对水平列表使用上述算法。
4. Go down by the height plus depth of the box (that is, starting at the upper left corner of the box) on the reference line, and continue processing vertically.
   沿着参考线向下移动盒子的高度加深度（即从盒子的左上角开始），并继续垂直处理。

Note that the above processes do not describe the construction of boxes. That would (for instance) involve for vertical boxes the insertion of baselineskip glue. Rather, it describes the way the components of a finished box are arranged in the output.

请注意，上述过程并不描述盒子的构造过程。这将涉及（例如）插入基线间距粘连的垂直盒子。相反，它描述了完成的盒子中组件的排列方式。

5.4.5 Boxes and negative glue 盒子和负粘连

Sometimes it is useful to have boxes overlapping instead of line up. An easy way to do this is to use negative glue. In horizontal mode

有时，盒子重叠而不是对齐是有用的。一种简单的方法是使用负粘连。在水平模式中，

{\dimen0=\wd8 \box8 \kern-\dimen0}

places box 8 without moving the current location.

可以将盒子 8 放置在当前位置而不移动它。

More versatile are the macros \llap and \rlap, defined as

更灵活的是宏 \llap 和 \rlap，定义如下：

\def\llap#1{\hbox to 0pt{\hss #1}}

and

\def\rlap#1{\hbox to 0pt{#1\hss}}

that allow material to protrude left or right from the current location. The \hss glue is equivalent to \hskip 0pt plus 1fil minus 1fil, which absorbs any positive or negative width of the argument of \llap or \rlap.

它们允许内容从当前位置向左或向右突出。\hss 粘连相当于 \hskip 0pt plus 1fil minus 1fil，可以吸收 \llap 或 \rlap 参数的任何正宽度或负宽度。

> The sequence
> \llap{\hbox to 10pt{a\hfil}}
> is effectively the same as 实际上等同于
> \hbox{\hskip-10pt \hbox to 10pt{a\hfil}}
> which has a total width of 0pt.
> 总宽度为 0 pt。

5.5 Overfull and underfull boxes

过充实和不充实的盒子

If a box has a size specification TEX will stretch or shrink glue in the box. For glue with only finite stretch or shrink components the badness (see Chapter ??) of stretching or shrinking is computed. In TEX version 3 the badness of the box most recently constructed is available for inspection by the user through the \badness parameter. Values for badness range 0–10 000, but if the box is overfull it is 1 000 000.

如果盒子具有尺寸规格，TEX 将拉伸或收缩盒子中的粘连。对于只有有限伸缩或收缩部分的粘连，计算其伸展或收缩的劣度/（详见第 ?? 章）。在 TEX 第 3 版中，用户可以通过 \badness 参数检查最近构造的盒子的劣度。劣度的值范围为 0–10, 000，但如果盒子过充实，则为 1, 000, 000。

When TEX considers the badness too large, it gives a diagnostic message. Let us first consider error reporting for horizontal boxes.

当 TEX 认为劣度太大时，它会给出诊断信息。首先让我们考虑水平盒子的错误报告。

Horizontal boxes of which the glue has to stretch are never reported if \hbadness $\geq$ 10 000; otherwise TEX reports them as 'underfull' if their badness is more than \hbadness.

如果 \hbadness $\geq 10,000$，则需要拉伸粘连的水平盒子永远不会报告；否则，如果它们的劣度大于 \hbadness，则 TEX 会将它们报告为“不充实”。

Glue shrinking can lead to 'overfull' boxes: a box is called overfull if the available shrink is less than the shrink necessary to meet the box specification. An overfull box is only reported if the difference in shrink is more than \hfuzz, or if \hbadness $<$ 100 (and it turns out that using all available shrinkability has badness 100).

粘连的收缩可能导致盒子“过充实”：如果可用的收缩量小于满足盒子规格所需的收缩量，则称为过充实的盒子。仅当收缩的差异大于 \hfuzz，或者如果 \hbadness $<$ 100（并且使用了所有可用的收缩性），过充实的盒子才会被报告。

Setting \hfuzz=1pt will let TEX ignore boxes that can not shrink enough if they lack less than 1pt. In

> 设置 \hfuzz=1pt 会使 TEX 忽略无法收缩足够的盒子，如果它们缺少的部分小于 1pt。在
>
> \hbox to 1pt{\hskip3pt minus .5pt}
>
> \hbox to 1pt{\hskip3pt minus 1.5pt}
>
> only the first box will give an error message: it is 1.5pt too big, whereas the second lacks .5pt which is less than \hfuzz.
>
> 中，只有第一个盒子会给出错误消息：它大了 1.5pt，而第二个盒子缺少 .5pt，这小于 \hfuzz。

Also, boxes that shrink but that are not overfull can be reported: if a box is 'tight', that is, if it uses at least half its shrinkability, TEX reports this fact if the computed badness (which is between 13 and 100) is more than \hbadness.

此外，可以报告收缩但未溢出的盒子：如果一个盒子是"紧凑的"，即它使用了至少一半的收缩能力，则当计算得到的劣度（介于 13 和 100 之间）大于 \hbadness 时，TEX 会报告这一事实。

For horizontal and vertical boxes this error reporting is almost the same, with parameters \vbadness and \vfuzz. The difference is that for horizontal overfull boxes TEX will draw a rule to the right of the box that has the same height as the box, and width \overfullrule. No overfull rule ensues if the \tabskip glue in an \halign cannot be shrunk enough.

对于水平和垂直盒子，这种错误报告几乎是相同的，使用的参数是 \vbadness 和 \vfuzz。不同之处在于，对于水平溢出的盒子，TEX 会在盒子右侧绘制一条与盒子高度相同且宽度为 \overfullrule 的标线。如果 \halign 中的 \tabskip 粘连无法缩小足够多，则不会出现溢出的标线。

5.6 Opening and closing boxes 打开和关闭盒子

The opening and closing braces of a box can be either explicit, that is, character tokens of category 1 and 2, or implicit, a control sequence \let to such a character. After the opening brace the \everyhbox or \everyvbox tokens are inserted. If this box appeared in a \setbox assignment any \afterassignment token is inserted even before the 'everybox' tokens.

盒子的左右花括号可以是显式的，也就是类别码为 1 和 2 的字符记号，或是隐式的，即 \let 为该类别码的字符的控制序列。在左花括号之后，会插入 \everyhbox 或 \everyvbox 记号。如果此盒子出现在 \setbox 赋值中，任何 \afterassignment 记号甚至会在 ‘everybox’ 记号之前插入。

```
\everyhbox{b}
\afterassignment a
\setbox0=\hbox{c}
\showbox0
gives
> \box0=
\hbox(6.94444+0.0)x15.27782
.\tenrm a
.\tenrm b
.\kern0.27779
.\tenrm c
```

Implicit braces can be used to let a box be opened or closed by a macro, for example:

隐式花括号可用于通过宏打开或关闭盒子，例如：

```
\def\openbox#1{\setbox#1=\hbox\bgroup}
\def\closebox#1{\egroup\DoSomethingWithBox#1}
\openbox0 ... \closebox0
```

This mechanism can be used to scoop up paragraphs:

这个机制可以用来收集段落：

```
\everypar{\setbox\parbox=
   \vbox\bgroup
       \everypar{}
       \def\par{\egroup\UseBox\parbox}}
```

Here the \everypar opens the box and lets the text be set in the box: starting for instance

这里的 \everypar 打开盒子并让文本在盒子中设置：例如从以下位置开始

```
Begin a text ...
```

gives the equivalent of

等同于

```
\setbox\parbox=\vbox{Begin a text ...
```

Inside the box \par has been redefined, so

在盒子内部，\par 被重新定义，因此

```
... a text ends.\par
```

is equivalent to

等同于

```
... a text ends.}\Usebox\parbox
```

In this example, the \UseBox command can only treat the box as a whole; if the elements of the box should somehow be treated separately another approach is necessary. In

在这个示例中，\UseBox 命令只能将整个盒子作为一个整体处理；如果盒子的元素需要以某种方式单独处理，就需要另一种方法。在

```
\everypar{\setbox\parbox=
  \vbox\bgroup\everypar{}%
       \def\par{\endgraf\HandleLines
                \egroup\box\parbox}}
\def\HandleLines{ ... \lastbox ... }
```

the macro \HandleLines can have access to successive elements from the vertical list of the paragraph. See also the example on page 1.

中，宏 \HandleLines 可以访问段落的垂直列表中的连续元素。还可以参考第 1 页上的示例。

5.7 Unboxing 拆箱

Boxes can be unwrapped by the commands \unhbox and \unvbox, and by their copying versions \unhcopy and \unvcopy. These are horizontal and vertical commands (see Chapter ??), considering that in effect they contribute a partial horizontal or vertical list. It is not possible to \unhbox a register containing a \vbox or vice versa, but a void box register can both be \unhboxed and \unvboxed.

可以使用命令 \unhbox 和 \unvbox 以及它们的拷贝版本 \unhcopy 和 \unvcopy 来拆开盒子。这些是水平和垂直命令（参见第 ?? 章），因为实际上它们贡献了一个部分的水平或垂直列表。不能对包含 \vbox 的寄存器进行 \unhbox，反之亦然，但空的盒子寄存器既可以进行 \unhbox，也可以进行 \unvbox。

Unboxing takes the contents of a box in a box register and appends them to the surrounding list; any glue can then be set anew. Thus

拆箱将盒子寄存器中盒子的内容添加到周围的列表中；然后可以重新设置任何粘连。因此，

\setbox0=\hbox to 1cm{\hfil} \hbox to 2cm{\unhbox0}

is completely equivalent to

完全等价于

\hbox to 2cm{\hfil}

and not to

，而不是

\hbox to 2cm{\kern1cm}

The intrinsically horizontal nature of \unhbox is used to define

\unhbox 的固有水平特性用于定义：

\def\leavevmode{\unhbox\voidb@x}

This command switches from vertical mode to horizontal without adding anything to the horizontal list. However, the subsequent \indent caused by this transition adds an indentation box. In horizontal mode the \leavevmode command has no effect. Note that here it is not necessary to use \unhcopy, because the register is empty anyhow.

此命令在不向水平列表中添加任何内容的情况下从垂直模式切换到水平模式。但是，由于此过渡引起的随后的 \indent 会添加一个缩进盒子。在水平模式下，

\leavevmode 命令没有任何效果。请注意，此处无需使用 \unhcopy，因为寄存器本身就是空的。

Beware of the following subtlety: unboxing in vertical mode does not add interline glue between the box contents and any preceding item. Also, the value of \prevdepth is not changed, so glue between the box contents and any following item will occur only if there was something preceding the box; interline glue will be based on the depth of that preceding item. Similarly, unboxing in horizontal mode does not influence the \spacefactor.

请注意以下微妙之处：在垂直模式下拆箱不会在盒子内容和任何前面的项目之间添加行间粘连。此外，\prevdepth 的值不会改变，因此只有在盒子之前有内容时，盒子内容与任何后续项目之间才会产生粘连；行间粘连将基于该前置项目的深度。类似地，在水平模式下拆箱不会影响 \spacefactor。

5.8 Text in boxes 盒子中的文本

Both horizontal and vertical boxes can contain text. However, the way text is treated differs. In horizontal boxes the text is placed in one straight line, and the width of the box is in principle the natural width of the text (and other items) contained in it. No ⟨vertical command⟩s are allowed inside a horizontal box, and \par does nothing in this case.

水平盒子和垂直盒子都可以包含文本。然而，文本的处理方式有所不同。在水平盒子中，文本被放置在一条直线上，盒子的宽度原则上是其中包含的文本（和其他项目）的自然宽度。水平盒子内不允许出现 ⟨vertical command⟩，且 \par 在这种情况下不起作用。

For vertical boxes the situation is radically different. As soon as a character, or any other ⟨horizontal command⟩ (see page ??), is encountered in a vertical box, TEX starts building a paragraph in unrestricted horizontal mode, that is, just as if the paragraph were directly part of the page. At the occurrence of a ⟨vertical command⟩ (see page ??), or at the end of the box, the paragraph is broken into lines using the current values of parameters such as \hsize.

对于垂直盒子，情况完全不同。当在垂直盒子中遇到字符或任何其他 ⟨horizontal command⟩（参见第 ??页）时，TEX 会开始在无限水平模式下构建段落，就好像该段落直接是页面的一部分。在遇到 ⟨vertical command⟩（参见第?? 页）或盒子的结尾时，使用诸如 \hsize 等参数的当前值将段落分成行。

Thus 因此，

\hbox to 3cm{\vbox{some reasonably long text}}

will not give a paragraph of width 3 centimetres (it gives an overfull horizontal box if \hsize > 3cm). However,

不会生成宽度为 3 厘米的段落（如果 \hsize > 3cm，则会得到一个过充实的水平盒子）。然而，

\vbox{\hsize=3cm some reasonably long text}

will be 3 centimetres wide.

将是 3 厘米宽。

A paragraph of text inside a vertical box is broken into lines, which are packed in horizontal boxes. These boxes are then stacked in internal vertical mode, possibly with \baselineskip and \lineskip separating them (this is treated in Chapter ??). This process is also used for text on the page; the boxes are then stacked in outer vertical mode.

垂直盒子中的文本段落会被分成多行，这些行会被放置在水平盒子中。然后，这些盒子会以内部垂直模式堆叠在一起，可能使用 \baselineskip 和 \lineskip 将它们分隔开（这在第 ?? 章中讨论）。这个过程也用于页面上的文本；然后这些盒子会在外部垂直模式中堆叠。

If the internal vertical list is empty, no \parskip glue is added at the start of a paragraph.

如果内部垂直列为空，则不会在段落开头添加 \parskip 粘连。

Because text in a horizontal box is not broken into lines, there is a further difference between text in restricted and unrestricted horizontal mode. In restricted horizontal mode no discretionary nodes and whatsit items changing the value of the current language are inserted. This may give problems if the text is subsequently unboxed to form part of a paragraph.

由于水平盒子中的文本不会被分成行，因此在受限制的和无限制的水平模式之间还有一个区别。在受限制的水平模式中，不插入离散节点和改变当前语言值的转换项。如果随后将文本解包以形成段落的一部分，则可能会出现问题。

See Chapter ?? for an explanation of these items, and [?] for a way around this problem.

有关这些项目的解释，请参见第 ?? 章，并参考 [?] 以解决此问题。

5.9 Assorted remarks 其他说明

5.9.1 Forgetting the \box 忘记 \box

After \newcount\foo, one can use \foo on its own to get the \foo counter. For boxes, however, one has to use \box\foo to get the \foo box. The reason for this is that there exists no separate \boxdef command, so \chardef is used (see Chapter ??).

在 \newcount\foo 之后，可以直接使用 \foo 得到 \foo 计数器。然而，对于盒子，必须使用 \box\foo 才能得到 \foo 盒子。原因是不存在单独的 \boxdef 命令，因此使用了 \chardef（见第 ?? 章）。

> Suppose \newbox\foo allocates box register 25; then typing \foo is equivalent to typing \char25.
> 假设 \newbox\foo 分配了盒子寄存器 25；那么输入 \foo 等同于输入 \char25.

5.9.2 Special-purpose boxes 特殊用途的盒子

Some box registers have a special purpose:

特殊用途的盒子

- \box255 is by used TEX internally to give the page to the output routine.
  \box255 是 TEX 在内部使用的，用于将页面传递给输出例程。
- \voidb@x is the number of a box register allocated in plain.tex; it is supposed to be empty always. It is used in the macro \leavevmode and others.
  \voidb@x 是在 plain.tex 中分配的盒子寄存器的编号；它应该始终为空。它在 \leavevmode 等宏中使用。
- when a new \insert is created with the plain TEX \newinsert macro, a \count, \dimen, \skip, and \box all with the same number are reserved for that insert. The numbers for these registers count down from 254.
  当使用 plain TEX 的 \newinsert 宏创建新的 \insert 时，一个具有相同编号的 \count、\dimen、\skip 和 \box 被保留给该插入。这些寄存器的编号从 254 开始递减。

5.9.3 The height of a vertical box in horizontal mode 在水平模式中的垂直盒子的高度

In horizontal mode a vertical box is placed with its reference point aligned vertically with the reference point of the surrounding box. TEX then traverses its contents starting at the left upper corner; that is, the point that lies above the reference point by a distance of the height of the box. Changing the height of the box implies then that the contents of the box are placed at a different height.

在水平模式中，垂直盒子与周围盒子的基准点垂直对齐。然后，TEX 从左上角开始遍历其内容；也就是说，该点与基准点之上的点的距离等于盒子的高度。改变盒子的高度意味着盒子的内容将以不同的高度放置。

Consider as an example

以以下代码为例：

```
\hbox{a\setbox0=\vbox{\hbox{b}}\box0 c}
```

which gives 将得到

abc

and 而

```
\hbox{a\setbox0=\vbox{\hbox{b}}\ht0=0cm \box0 c}
```

which gives 将得到

a$_\text{b}$c

By contrast, changing the width of a box placed in vertical mode has no effect on its placement.

相比之下，更改放置在垂直模式中的盒子的宽度对其放置没有影响。

5.9.4 More subtleties with vertical boxes 垂直盒子的更多细节

Since there are two kinds of vertical boxes, the \vbox and the \vtop, using these two kinds nested may lead to confusing results. For instance,

由于有两种类型的垂直盒子，\vbox 和 \vtop，在嵌套使用这两种类型时可能会导致混淆的结果。例如，

```
\vtop{\vbox{...}}
```

is completely equivalent to just

完全等效于只是

```
\vbox{...}
```

It was stated above that the depth of a \vbox is zero if the last item is a kern or glue, and the height of a \vtop is zero unless the first item in it is a box. The above examples used a kern for that first or last item, but if, in the case of a \vtop, this item is not a glue or kern, one is apt to overlook the effect that it has on the surrounding box. For instance,

前面已经提到，如果最后一个项目是间距（glue）或紧排（kern），则\vbox 的深度为零，而\vtop 的高度为零，除非其第一个项目是一个盒子。前面的示例中使用了间距或紧排作为第一个或最后一个项目，但是如果在\vtop 的情况下，这个项目不是间距或紧排，人们很容易忽视它对周围盒子的影响。例如，

```
\vtop{\write16{...}...}
```

has zero height, because the write instruction is packed into a 'whatsit' item that is placed on the current, that is, the vertical, list. The remedy here is

的高度为零，因为写入指令被打包为一个“whatsit”项目，该项目被放置在当前（也就是垂直）列表中。此处的解决方法是

```
\vtop{\leavevmode\write16{...}...}
```

which puts the whatsit in the beginning of the paragraph, instead of above it.

它将 whatsit 放在段落的开头，而不是在段落之上。

Placement of items in a vertical list is sometimes a bit tricky. There is for instance a difference between how vertical and horizontal boxes are treated in a vertical list. Consider the following examples. After \offinterlineskip the first example

在垂直列表中放置项目有时会有些棘手。例如，垂直盒子和水平盒子在垂直列表中的处理方式有所不同。考虑以下示例。在执行\offinterlineskip 之后，第一个示例

```
\vbox{\hbox{a}
    \setbox0=\vbox{\hbox{(}}
    \ht0=0pt \dp0=0pt \box0
    \hbox{ b}}
```

gives

```
    a
    (b
```

while a slight variant

而稍微有些变化的第二个示例

```
\vbox{\hbox{a}
    \setbox0=\hbox{(}
    \ht0=0pt \dp0=0pt \box0
    \hbox{ b}}
```

gives

```
    (a
     b
```

The difference is caused by the fact that horizontal boxes are placed with respect to their reference point, but vertical boxes with respect to their upper left corner.

这种差异是由于水平盒子相对于它们的参考点进行定位，而垂直盒子相对于它们的左上角进行定位。

5.9.5 Hanging the \lastbox back in the list 将 \lastbox 挂回列表

You can pick the last box off a vertical list that has been compiled in (internal) vertical mode. However, if you try to hang it back in the list the vertical spacing may go haywire. If you just hang it back,

您可以从已经在（内部）垂直模式下编译的垂直列表中取出最后一个盒子。然而，如果您试图将其挂回到列表中，垂直间距可能会出现问题。如果只是简单地将其挂回，

```
\setbox\tmpbox=\lastbox
\usethetmpbox \box\tmpbox
```

baselineskip glue is added a second time. If you 'unskip' prior to hanging the box back,

基线间距粘连会被添加两次。如果在挂回盒子之前进行“去除间距”操作，

```
\setbox\tmpbox=\lastbox \unskip
\usethetmpbox \box\tmpbox
```

things go wrong in a more subtle way. The ⟨internal dimen⟩ \prevdepth (which controls interline glue; see Chapter ??) will have a value based on the last box, but what you need for the proper interline glue is a depth based on one box earlier. The solution is not to unskip, but to specify \nointerlineskip:

会以更微妙的方式出现问题。⟨ 内部尺寸 ⟩ \prevdepth（控制行间粘连；参见第 ?? 章）将基于最后一个盒子的值，但是对于正确的行间粘连所需的深度应该基于一个盒子较早的值。解决方法不是去除间距，而是指定 \nointerlineskip：

```
\setbox\tmpbox=\lastbox
\usethetmpbox \nointerlineskip \box\tmpbox
```

5.9.6 Dissecting paragraphs with \lastbox 使用 \lastbox 分解段落

Repeatedly applying \last... and \un... macros can be used to take a paragraph apart. Here is an example of that.

可以使用重复应用 \last... 和 \un... 宏的方法来分解段落。以下是一个示例。

In typesetting advertisement copy, a way of justifying paragraphs has become popular in recent years that is somewhere between flushright and raggedright setting. Lines that would stretch beyond certain limits are set with their glue at natural width. This paragraph exemplifies this procedure; the macros follow next.

在排版广告文案时，近年来流行一种介于右对齐和左对齐之间的段落对齐方式。会超过一定限制的行会以自然宽度设置其粘连。本段落展示了这个过程；下面是相关的宏。

```
\newbox\linebox \newbox\snapbox
\def\eatlines{
   \setbox\linebox\lastbox    % check the last line % 检查最后一行
   \ifvoid\linebox
   \else                      % if it's not empty
   \unskip\unpenalty          % take whatever is
   {\eatlines}                % above it;
                              % collapse the line
   \setbox\snapbox\hbox{\unhcopy\linebox}
                        % depending on the difference
   \ifdim\wd\snapbox<.98\wd\linebox
       \box\snapbox % take the one or the other,
   \else \box\linebox \fi
   \fi}
```

This macro can be called as

可以使用以下方式调用此宏：

```
\vbox{ ... some text ... \par\eatlines}
```

or it can be inserted automatically with \everypar; see [?].

或者可以使用 \everypar 自动插入；参见 [?]。

In the macro \eatlines, the \lastbox is taken from a vertical list. If the list is empty the last box will test true on \ifvoid. These boxes containing lines from a paragraph are actually horizontal boxes: the test \ifhbox applied to them would give a true result.

在宏 \eatlines 中，\lastbox 是从垂直列表中取出的。如果列表为空，则最后一个盒子在 \ifvoid 上测试为真。这些包含段落中的行的盒子实际上是水平盒子：对它们应用 \ifhbox 测试将得到真结果。