



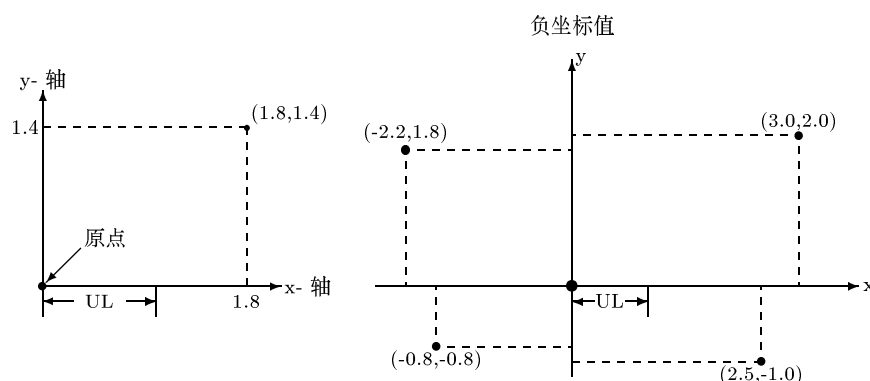
## 第六章 图形

利用  $\text{\LaTeX}$  可以生成简单的图形和插图。生成图形的构造模块是文本、各种斜率的直线、箭头、圆、卵形线和线段等等，用户可以把这些模块放在任何所希望的地方。

### §6.1 图形的尺寸和位置

只有为图形建立了一个坐标系，其构造模块才有可能被安置。而坐标系由参考点 (或原点) 和两条互相垂直的坐标轴组成，坐标轴上要有单位长度以确定坐标。这里所取的原点就是图形的左下角，坐标轴就是底边与左边。这些边称为  $x$ -轴 (底边) 和  $y$ -轴 (左边)。

一旦定义了单位长度 (UL)，图形中每点都唯一被两个数确定：第一个是沿  $x$ -轴的长度，第二个是沿  $y$ -轴的长度。



单位长度的选择是用如下命令进行的：

```
\setlength{\unitlength}{长度}
```

在上图中左边那个例子，单位长度用命令 `\setlength{\unitlength}{1.5cm}` 设置成 1.5cm。那么点 (1.8, 1.4) 即是指位于原点右边 1.8 倍单位长度 (= 2.7cm)，原点上边 1.4 倍单位长度 (= 2.1cm)。

坐标值一般都是正数，这就是说点位于参考点的右上方向。由于参考点是图形的左下角，因此所有点都应该在它的右上方向。然而，坐标也可以取负值。负的  $x$  值 (坐标对中第一个数为负数) 定义的点在原点的左边，而负的  $y$  值 (坐标对中第二个数为负数) 定义的点在原点的下边。

上图右边的例子说明了这种情形。这里的单位长度为 1cm，因此坐标值就是两个方向上离原点，以 cm 为单位的距离。

单位长度通常就设为方便的尺寸，如 1cm, 1mm 或 1in，然后相应地建立我们的图形。当整幅图形完成后，只需修改单位长度的定义，就可以放缩

图形。一幅原始设计时 `\unitlength` 取 1cm，可以通过把单位长度重定义为 1.2cm 来放在为原来的 1.2 倍。

## §6.2 图形环境

图形是用 `picture` 环境构造的，其语法为

```
\begin{picture}(x 尺寸, y 尺寸)
```

画图命令

```
\end{picture}
```

这里的 (x 尺寸, y 尺寸) 是一组数，它定义了图形在 x- 方向（水平）和 y- 方向（竖直）的尺寸（范围）。这对数是用小括号围起来的！单位长度就是在此之前所定义的 `\unitlength`。

```
\setlength{\unitlength}{1.5cm}
```

```
\begin{picture}(4,5) ... .. \end{picture}
```

的结果为一幅 4 倍单位长度宽，5 倍单位长度高的图形。而由于已把单位长度设为 1.5cm，因此图形的实际尺寸为宽 6cm，高 7.5cm。

而画图命令就是下面将要介绍的用以生成和定位各个图形要素的命令。这些命令再加上字体样式和尺寸声明（4.1 节）以及线粗命令 `\thincklines` 与 `\thinlines` 就是可以位于 `picture` 环境中的所有命令。后面这两条命令用来确定两种可用线粗中的哪一种被用来绘制当前直线：我们按自己的意愿在两者之间来回切换。刚开始时激活的是细线。

参数 `\unitlength` 的值一定不能在 `picture` 环境内改变，因为对它而言，单位长度必须维持不变。当然可以在两幅图形中间改变它的值。

如果 `\unitlength` 的定义随同 `picture` 环境一起被包围在另一个类似于 `\begin{center} ... \end{center}` 这样的环境内，那么 `\unitlength` 值的作用持续到这个环境结束。一个前面没有 `\unitlength` 定义的 `picture` 环境，就会采用它的标准值 1pt。

## §6.3 定位命令

图形中的元素是用两条命令 `\put` 和 `\multiput` 来创建和定位的，它们的语法为：

```
\put(x- 坐标, y- 坐标){图形元素}
```

```
\multiput(x- 坐标, y- 坐标)(x- 增量, y- 增量){数}{图形元素}
```

这里的 图形元素 就是下一节要讲述的画形基本命令。参数值 (x-坐标, y-坐标) 是安置坐标，规定了元素在画形坐标系中的位置，它是以 `\unitlength` 为

单位。如果单位长度为 1cm，那么 (2.5, 3.6) 就意味着元素定位在相对于图形左下角向右 2.5cm，向上 3.6cm 的地方。

命令 `\multiput` 是把同样的图形元素生成数次，每次移动 (x-增量, y-增量)。因此元素被重复地画在

(x-坐标, y-坐标), (x-坐标+x-增量, y-坐标+y-增量),  
 (x-坐标+2x-增量, y-坐标+2y-增量), ..., 一直到  
 (x-坐标+[数-1]x-增量, y-增量+[数-1]y-增量)

每画一次，(x-坐标, y-坐标) 都要增加 (x-增量, y-增量)。这里被增加的量可以是正数，也可以是负数。

因此 `\multiput(2.5, 3.6)(0.5, -0.6){5}{图形元素}` 就会画五次图形元素，第一个的位置在 (2.5, 3.6)，然后是 (3.0, 3.0), (3.5, 2.4), (4.0, 1.8)，最后是 (4.5, 1.2)。

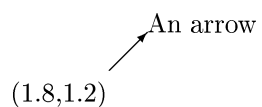
注意坐标和增量数对都是放在小括号 ( , ) 内，其中的两个数是用逗号分开的。而数和图形元素项则是像通常那样放在大括号内。

警告：由于这里是用逗号分开两个数，因此不能再用它表示小数点。对于坐标项，小数点必须是句号，而不能是逗号。

## §6.4 基本画图命令

### §6.4.1 图形中的文本

所有画形元素中最简单的就是一段文本，定位在图形中需要的位置。只要把文本放在 `\put` 或 `\multiput` 命令中的图形元素所在地方就可以了。

 An arrow 箭头所指的位置为 (1.8, 1.2)。  
 利用命令 `\put(1.8, 1.2){An arrow}` 就会插入文本  
 ‘An arrow’，其左下角就是所指的位置。

做为图形中元素的文本也可以包装进一个 `\parbox` 或 `minipage` 环境中，这时在 `\put` 命令中坐标项的参考点与竖直盒子的定位参数值有关：

<code>\parbox[b]{...}{...}</code>	<code>\parbox{32cm}{...}</code>	<code>\parbox[t]{...}{...}</code>
Reference point is the lower left corner of the last line in the parbox	For a standard parbox, the reference point is the vertical center of the left edge	Reference point is the lower left corner of the top line in the parbox

**练习 6.1:** 生成一幅宽 100mm，高 50mm 的图形，这里 `\unitlength` 取值 1mm。把给定文本放在下列位置：(0,0) ‘The First Picture’, (90, 47) ‘upper left’, (70,40) ‘somewhere upper right’，并且放一个宽为 60mm 的子段盒子

在 (25,25) 点处, 内容是 ‘A separate exercise file with the name `picture.tex` should be created for the exercise in this Chapter.’

**练习 6.2:** 给 `\unitlength` 取值 1.5mm, 再重复上述的绘制操作, 并且在子段盒子中让定位参数值分别取 `t` 和 `b`。

### §6.4.2 图形中的盒子 — 矩形

在 `picture` 环境中也可用盒子命令 `\framebox`, `\makebox` 和 `\savebox` (4.7.1 节), 但是其语法已做了推广。而且, 还有一个盒子命令 `\dashbox`:

`\makebox(x-尺寸, y-尺寸)[位置]{文本}`

`\framebox(x-尺寸, y-尺寸)[pos]{text}`

`\dashbox{虚线尺寸}(x-尺寸, y-尺寸)[位置]{文本}`

尺寸数对 (x-尺寸, y-尺寸) 定义了矩形的宽度和高度, 它们以 `\unitlength` 为单位。定位参数值 `位置` 定义了文本在盒子中的位置。它可以取如下值:

[t] top— 输入文本水平居中地位于盒子顶边的下面。

[b] bottom— 输入文本水平居中地位于盒子底边的上面。

[l] left— 输入文本竖直居中地位于盒子的左边。

[r] right— 输入文本竖直居中地位于盒子的右边。

<sup>2ε</sup>[s] stretch— 输入文本竖直居中, 但要水平伸展以充满整个盒子。

如果没有可省参数 `位置`, 那么输入文本是水平竖直居中放置在盒子中。

也可以一次这些参数中的两个组合起来使用:

[tl] top left— 输入文本位于左上角。

[tr] top right— 输入文本位于右上角。

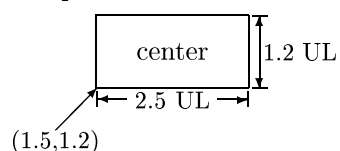
[bl] bottom left— 输入文本位于左下角。

[br] bottom right— 输入文本位于右下角。

这里值的顺序是无关紧要的, `tl` 与 `lt` 的效果相同。

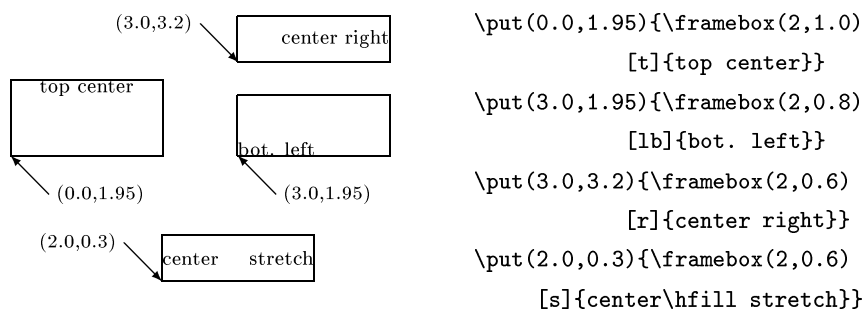
这些命令就是用在 `\put` 和 `\multiput` 命令中图形元素处。盒子的放置方式是其左下角所处位置就是放置命令中的坐标对。

`\put(1.5,1.2){\framebox(2.5,1.2){center}}`

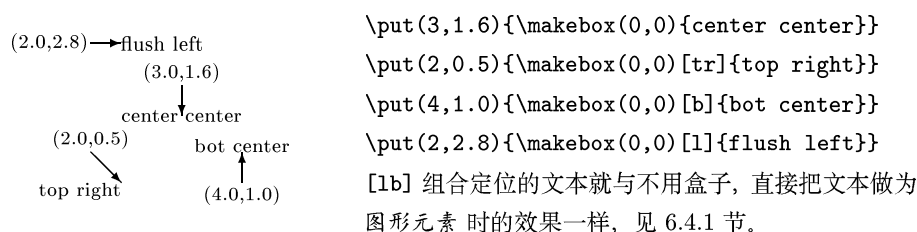


箭头所指位置为 (1.5,1.2), 这就是宽 2.5 单位, 高 1.2 单位的矩形左下角所在的地方。文本 ‘center’ 是水平和竖直居中的。  
UL = 0.8cm。

下面这个例子更好地说明了文本定位参数值的效果 (UL = 1cm):

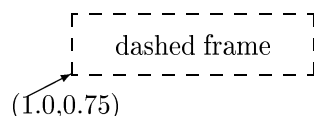


图形元素 `\makebox` 同 `\framebox` 命令完全一样，只是它没有矩形框而已。对它而言，经常把范围对取为 (0,0)，这样可以把文本放在所希望的地方。（关于零宽度盒子对被包围文本的效果请见 4.7.1 节。）



图形元素 `\dashbox` 也生成有框盒子，不过其框线为虚线。参数值 虚线尺寸 就是用来定义短线长度。

```
\put(1.0,0.75){\dashbox{0.2}(4,1){dashed frame}}
```



当盒子的宽度和高度都是短线长度的倍数时，虚线框要好看一些。

即使上面这些图形盒子命令中，也可以把文本放在竖直盒子 (`\parbox` 或 `minipage`) 中。由于竖直盒子自身具有可省的定位参数值 `b` 或 `t`，它一定不能与图形盒子的定位参数值冲突，因此要遵守下面的规则：

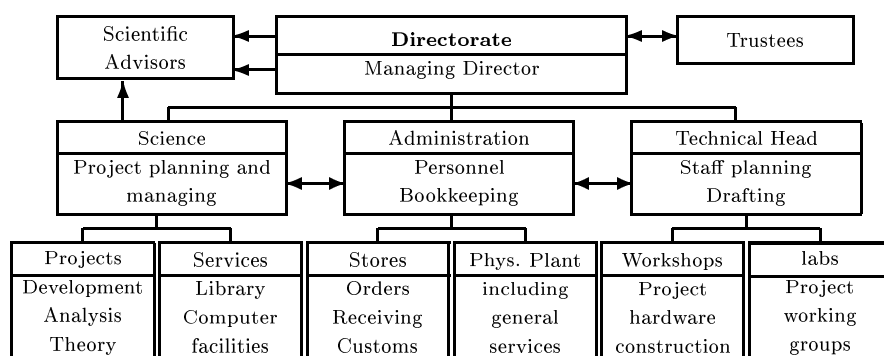
如果图形盒子中包含了定位参数值 `b` 或 `t`，那么被包围的竖直盒子中也必须有相同的定位参数值。如果图形盒子中没有定位参数值，或者只是 `r` 或 `l`，那么竖直盒子必须是标准（无参数值）形式。

图形盒子中的定位参数值对于被包围竖直盒子的作用同它对一行文本的作用一样，都是把它们当做一个整体对待。

**练习 6.3:** 复制下面这个机构表格，要求包含文本，但现在先不生成水平和竖直直线及箭头，它们是后面要做的练习。

提示：首先在一张有格的纸上画出方框，而且盒子的边与格线重合。把单位长度就取为格线间距。把原点取做想像中的包含所有盒子的方框左下角。

注意：很快你就能生成自己的有格子的页面，其中格线间距可以是任意希望的尺寸。



### §6.4.3 直线

在 `picture` 环境中, `LaTeX` 可以绘制任意长度的水平, 竖直以及有限倾角的直线。这个图形元素的语法是:

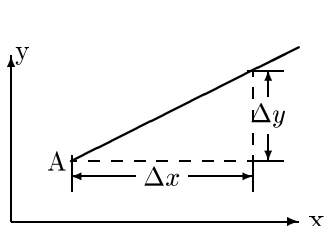
`\line( $\Delta x$ ,  $\Delta y$ )`{ 长度 }

对于水平线和竖直线, 长度 定义了以单位长度为单位的线长。对于其它倾角的直线, 它的意义就有点儿复杂, 稍后加以解释。直线开始于由 `\put` 或 `\multiput` 命令中给出的安置坐标确定的点。

```

\thicklines
\put(0,0){\line(1,0){6}}
\put(0,0){\line(0,1){1}}
\put(6,0){\line(0,1){0.5}}
```

绘制直线时的倾角是由斜率对  $(\Delta x, \Delta y)$  给定的。斜率对  $(1,0)$  中  $\Delta x = 1$ ,  $\Delta y = 0$ , 这样生成水平线, 而  $(0,1)$  生成的则是竖直线。上面的例子说明了这一点。一般地  $(\Delta x, \Delta y)$  具有下面的含义:



从直线上的 A 点开始, 沿着 x 方向 (水平) 走  $\Delta x$  距离, 那么  $\Delta y$  就是沿 y 方向 (竖直) 移动的距离, 从而可以重回到直线上。

通过定义斜率对  $(\Delta x, \Delta y)$ , 那么绘制出来的直线倾角应该满足上面的条件。

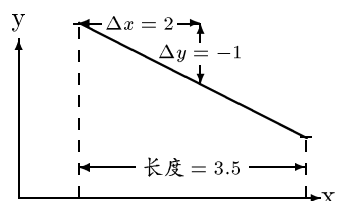
前面已经说过, 只可以绘制有效倾角的直线。这是因为  $\Delta x$  和  $\Delta y$  的取值要遵从下面的规则:

1. 数值必须是整数 (负数或正数均可)。
2. 只可以取值  $0, 1, \dots, 6$ 。
3. 在数对中的两数不能有公因子。

因此类似  $(3.5, 1.2)$  (规则 1) 和  $(7, 0)$  (规则 2) 这样的数对是不允许出现的。同样  $(2, 2)$  和  $(3, 6)$  不符合规则 3, 因为前一对中的两个数可以都被 2 整除, 而后一对中的两数可以都被 3 整除。可以用  $(1, 1)$  和  $(1, 2)$  来得到同

样的倾角。因此这里一共有 25 种可接受的斜率对，其中 (1,0) 和 (0,1) 分别相应于水平线和竖直线。可以通过把所有的可能写出来以验证这一总数。

另外，在斜率对中的数值可以是负数，也可以是正数，例如 (0,-1) 和 (-2,-5) 也是允许的。在上面图示中，负的  $\Delta x$  意味着向左移动，而负的  $\Delta y$  意味着向下移动。因此 `\put(2,3){\line(0,-1){2.5}}` 的结果是一条开始于 (2,3) 的直线，其竖直向下伸展长达 2.5 单位。



对于有倾角的直线，参数值 长度 定义的是沿 x-轴的投影长度。这一点可以借助于左边的图示更清楚地看出来。

`\put(1.0,2.75){\line(2,-1){3.5}}`

如果从两个端点竖直向下画虚线，那么在这两虚线之间的 x-轴部分就是直线在 x-轴上的投影。

倾斜直线的长度必须不能短于 10pt 或者 3.5mm，否则不会生成任何结果。但是如果包含了 pict2e 软件包 (6.5.6 节)，就不会有这种限制。

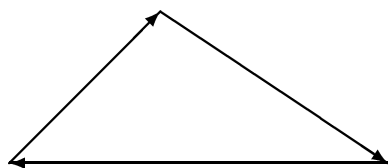
#### §6.4.4 箭头

箭头图形要素是用下面的命令生成的：

`\vector(Δx, Δy){长度}`

其作用方式同 `\line` 命令完全一样，而且参数值和其局限性也是相同的。这条命令从由 `\put` 或 `\multiput` 命令定义的位置开始画一条直线，然后在终点处画上箭头。

同直线一样，箭头的长度也不能少于 10pt 或 3.5mm。规则 1-3 也同样适用于  $\Delta x$  和  $\Delta y$ ，而且更进一步，要求可以取的值只能是 0,1,2,3,4。这样当不考虑正负号时，只能画 13 种不同倾角的箭头。



```
\begin{picture}(5,2)\thicklines
\put(5,0){\vector(-1,0){5}}
\put(0,0){\vector(1,1){2}}
\put(2,2){\vector(3,-2){3}}
\end{picture}
```

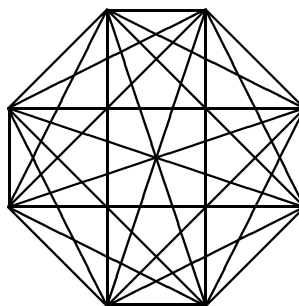
**练习 6.4:** 补上练习 6.3 中还没画的水平和竖直直线与箭头，完成那个演示图。

**练习 6.5:** 生成一张 6.5 × 9 英寸的方格纸，格线距离为 0.1in。这只需要用两个 `\multiput` 命令就可以了。在这层网格上重叠同样全局尺寸，但是格线距离为 0.5in 的方格，而且线粗由 `\thicklines` 给出。



**练习 6.6:** 生成右边的插图。

顶点是 (0,5), (0,10), (5,15), (10, 15), (15,10), (15,5), (10,0) 和 (5,0) , 单位长度为 0.1in 。



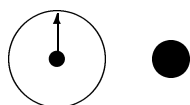
### §6.4.5 圆

圆这种图形要素是用下面的命令得到的:

```
\circle{ 直径 }
```

```
\circle*{ 直径 }
```

利用 \*- 形式的命令, 可以画出内部被填充了的实心圆, 而不是标准形式画出的那种轮廓线。只能画特定尺寸的圆, 因此 L<sup>A</sup>T<sub>E</sub>X 会选出与指定 直径 最接近的圆。( 6.5.6 节介绍的 `pict2e` 软件包可以绘制任意尺寸的圆。)



```
\begin{picture}(3,1.6)
\put(1,1){\circle*{0.2}}
\put(1,1){\circle{1.2}}
\put(1,1){\vector(0,1){0.6}}
\put(2.5,1){\vector*{0.5}}
\end{picture}
```

在相应的 `\put` 命令中的安置位置对应于圆心。

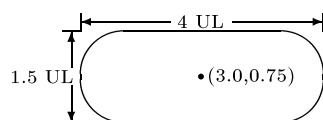
### §6.4.6 卵形线与圆角

我们这里所说的卵形线指的是一种矩形, 其顶角用四分之一圆角代替; 这里对直径的选取是使所有边光滑拼接的最大值。生成卵形线的命令是

```
\oval(x- 尺寸, y- 尺寸)[ 部分 ]
```

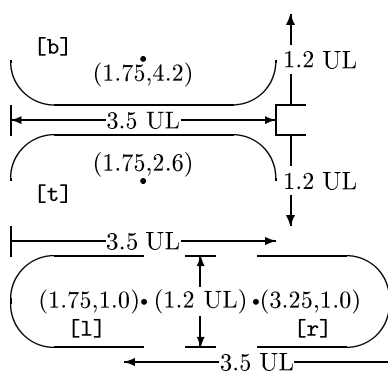
在相应 `\put` 命令中的安置坐标对应于卵形线的中心。

```
\put(3.0,0.75){\oval(4.0,1.5)}
```



这里我们取 x-尺寸=4.0 UL, y-尺寸=1.5 UL, 而单位长度 UL 已选择为 0.8cm。卵形线的中心就是 `\put` 命令中的安置坐标 (3.0,0.75)。

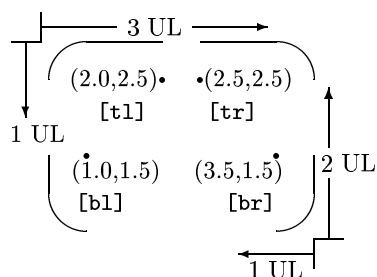
可省参数 部分 可以取值 `t`, `b`, `l` 或者 `r`, 以生成一半的卵形线。



```
\put(1.75,4.2){\oval(3.5,1.2)[b]}
\put(1.75,2.6){\oval(3.5,1.2)[t]}
\put(1.75,1.0){\oval(3.5,1.2)[l]}
\put(3.25,1.0){\oval(3.5,1.2)[r]}
```

半卵形线的宽度和高度与整个都要画出来时是一样的,即使这里只是画一半出来。类似地,在相应 `\put` 命令中的安置坐标仍然对应于完整卵形线的中心。(这里的单位长度为  $UL=1\text{ cm}$ 。)

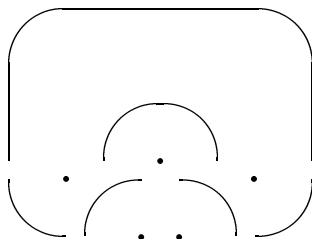
参数值 部分 也可以是四种组合 `tl`, `tr`, `bl` 或 `br` 中的一种,以生成四分之一的卵形线。这里两字母的顺序是无关紧要的,因此也可以用 `lt`, `rt`, `lb` 或 `rb`。



```
\put(2.0,2.5){\oval(3.0,1.0)[tl]}
\put(2.5,2.5){\oval(3.0,1.0)[tr]}
\put(1.0,1.5){\oval(1.0,2.0)[bl]}
\put(3.5,1.5){\oval(1.0,2.0)[br]}
```

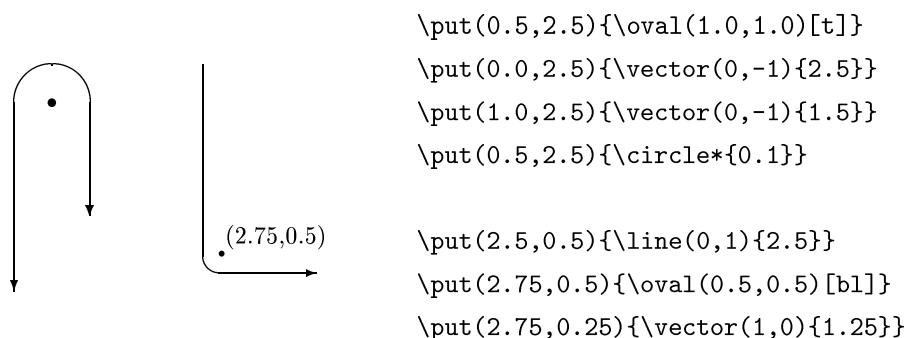
同样这里的尺寸定义仍旧参照完整卵形线而进行,即使画出来的只是一部分,在 `\put` 命令中的安置坐标也是指的整个卵形线的中心。

通过把卵形线中宽度与高度取成相等的值,可以得到四分之一或者半个圆周,但正如前面对圆的限制一样,也只能用特定的尺寸。下面的示例说明了部分圆周可以有小至  $1.5\text{cm}$  的尺寸。



```
\put(2.0,1.0){\oval(4.0,4.0)[tl]}
\put(2.0,1.0){\oval(1.5,1.5)[tr]}
\put(0.75,0.75){\oval(1.5,1.5)[bl]}
\put(1.75,0.0){\oval(1.5,1.5)[tl]}
\put(2.25,0.0){\oval(1.5,1.5)[tr]}
\put(3.25,0.75){\oval(1.5,1.5)[br]}
```

部分卵形线也可以与其它图形要素组合。当然这时可能需要对 `\put` 命令中的安置坐标进行相当仔细的考虑,以定位准确。

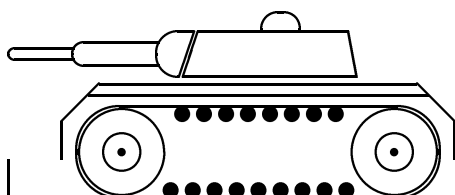


在上面所有例子中, 卵形线的中心都用一个点标记出来, 以说明其位置。通常它们并不是 `\oval` 图形要素的一部分。

**练习 6.7:** 虽然在冷战后年代, 画下面这个东西并不是很有意义, 但是它是一个相当棒的练习 `picture` 环境的机会。

提示: 用透明格纸罩在下面的图上, 可以很方便地进行定位和确定尺寸。

UL=1mm



### §6.4.7 竖直堆积文本

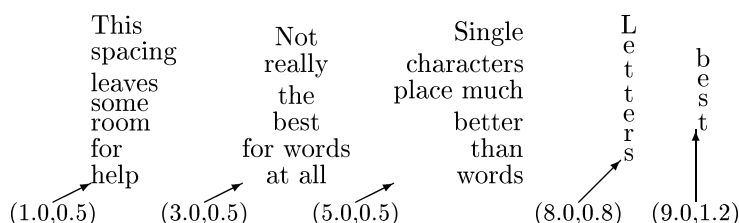
有时在演示图中需要沿竖直方向书写文本, 如本段右页那样所示。这可以用如下命令来做到

`\shortstack[位置]{列}`

位置 参数值可以取 `l`, `r` 或 `c`。标准值是 `c`, 表示居中。该命令类似于只有一列的 `tabular` 环境。列 表示输入文本, 行与行之间用 `\\` 分开。

y  
-  
a  
x  
i  
s

`\shortstack` 命令经常用来实现把单个字母上下安置, 即使是长文本也可以竖直堆积。行与行之间用尽可能小的间距分开。这就意味着没有向下或向上突出字母 (如 `h` 和 `y`) 的行离相邻行的距离要比有这样字母的行离相邻行近。



相应 `\put` 命令中的安置坐标对应于想像中的包含竖直堆积文本的盒子的左下角。上面例子中第一个的文本是左对齐的，第二个是居中，第三个是右对齐。最右面的两个是居中的。它们是用如下输入生成的：

```
\put(1.0,0.5){\shortstack[l]{This\spacing\leaves\some\ ...}}
\put(3.0,0.5){\shortstack{Not\really\the\best\ ...}}
\put(5.0,0.5){\shortstack[r]{Single\characters\ ...}}
\put(8.0,0.8){\shortstack{L\le\text\le\right\}}
\put(9.0,1.2){\shortstack{b\le\se\text}}
```

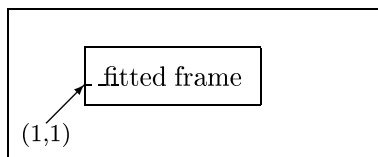
`\shortstack` 命令也可以用在 `picture` 环境外面的普通文本中。其中一个应用就是页边注，见 4.10.6 节。

### §6.4.8 有框文本

`\framebox` 命令生成一个具有预先定义尺寸的有框盒子，其中的文本可以放在不同的地方（6.4.2 节）。在文本模式中，命令 `\fbox` 可以围绕文本画一个方框，而且其会与文本匹配得很好（4.7.1 节）。这条命令也可以用在 `picture` 环境中。

在盒子的框线与被包围文本之间的距离是由参数 `\fboxsep` 所确定的。利用 `\put` 命令来放置一个 `\fbox`，其方式是出人意料的，请看下图：

```
\begin{picture}(5,2)
\setlength{\fboxsep}{0.25cm}
\put(0,0){\framebox(5,2){}}
\put(1,1){\fbox{fitted frame}}
\end{picture}
```

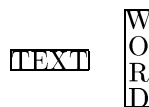


在演示图中通常不期望在框中出现多余的空白，特别当方框包围的是一幅图，而不是文本时更是如此。在这种情况下，可以使用命令

`\frame{图形元素}`

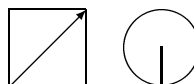
`\put` 命令中的安置坐标与通常一样，相应于左下角。

```
\put(0.0,0.5){\frame{TEXT}}
\put(1.5,0.0){\frame{\shortstack{W\O\R\D}}}
```



`\frame` 命令中的内容并不一定只是文本，也可以是前面给出的图形元素。然而，很多情形中输出总有点儿不对头。

```
\put(0,0){\frame{\vector(1,1){1.0}}}
\put(2,0){\frame{\circle{1.0}}}
```



第一条命令得到正确的结果，而第二条命令却失败了。对此情况，我们可以尝试把图形对象放在适当尺寸 `\makebox` 中，然后把它作在 `\frame` 命令的参数值。然而，用 `\framebox` 命令代替 `\frame{\makebox...}` 会更合理。

### §6.4.9 曲线

在 `picture` 环境中也可以用下面的命令画曲线：

`\bezier{ 数 }(x_1,y_1)(x_2,y_2)(x_3,y_3)`

$\overset{2\varepsilon}{\square}$  `\qbezier[ 数 ](x_1,y_1)(x_2,y_2)(x_3,y_3)`

它会画出一条从点  $(x_1, y_1)$  到  $(x_3, y_3)$  的二次 Bézier 曲线，而  $(x_2, y_2)$  是 Bézier 控制点。曲线实际上是用 数+1 个点画出来的。`\bezier` 和 `\qbezier` 命令之间的唯一区别在于，对后者而言，数是一个可省参数值；如果省略了它，那么就会计算生成一条光滑曲线所需要的点数。之所以还保留 `\bezier` 命令，就是为了与老的 L<sup>A</sup>T<sub>E</sub>X2.09 文档兼容。

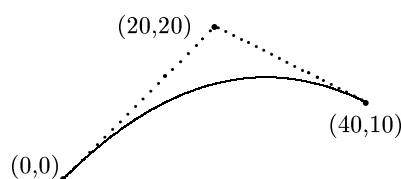
Bézier 控制点的意义可以从右面的

例子看出来。输入文本为

`\begin{picture}(40,20)`

`\qbezier(0,0)(20,20)(40,10)`

`\end{picture}`



这条曲线是从  $(0,0)$  到  $(40,10)$ ，那么在端点处的切线(虚线)就经过 Bézier 控制点  $(20,20)$ 。另一种表述这一点的方法是：当要从第一个点移到第三个点时，我们首先面向第二个点前进，为了到达目的地，我们要背向第二个点前进。上面例子中的虚线并不是 `\qbezier` 函数绘制出来的，只是为了说明上述含义。

通过指定点数，可以绘制出虚线，要得到这种效果，可能需要试验几次。

注意：`\bezier` 命令并不是积成在 L<sup>A</sup>T<sub>E</sub>X2.09 中的部分，它是包含在一个叫 `bezier.sty` 的文件中，必须把 `bezier` 列在 `\documentstyle` 的选项中，这样才能读取它。在 L<sup>A</sup>T<sub>E</sub>X 2<sub>ε</sub> 中为了保证兼容性，提供了同名的空文本。

## §6.5 其它图形命令与示例

### §6.5.1 直线粗细

对于图形元素 `\circle`、`\oval`、`\vector` 和斜线，存在两种可选择的线粗。即可以用

`\thicklines` 或 `\thinlines`

来选择粗线或细线。这两条命令的作用直到有相反命令为止，或者其所在环境结束。刚开始时 `\thinlines` 起作用。

水平或竖直直线的线粗可以用下面的声明来定义成任意期望的尺寸：

`\linethickness{ 粗细 }`

参数值 粗细 是正的长度定义。利用 `\linethickness{1.5mm}` 可以使得所有后面的水平或竖直线粗 1.5mm。如果包含了 `pict2e` 软件包 ( 6.5.6 节)，这

个粗细定义也同样适用于斜线。

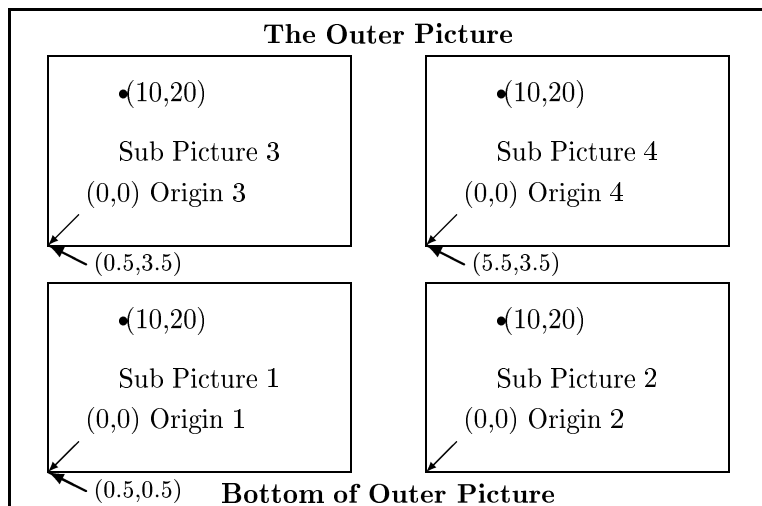
由于方框就是用竖直线和水平线构成的，因此命令 `\linethickness` 的定义对 `\framebox` 和 `\dashbox` 也同样有作用。

### §6.5.2 嵌套图形

在 `\put` 或 `\multiput` 命令中的图形元素也可以是另一个 `picture` 环境。如此多重图形的语法是：

```
\put(x- 坐标, y- 坐标){\setlength{\unitlength}{单位长度}
\begin{picture}(x- 尺寸, y- 尺寸)
... 子图形 ... \end{picture} }
```

在内部图形环境中的安置坐标是相对于它自己的原点的，即其左下角；而原点就对应于外面 `picture` 环境中 `\put` 命令的安置坐标。可以给内部 `picture` 环境选择不同的 `\unitlength` 单位长度值；然而，如果没有定义新的值，那么它就与外面 `picture` 环境相同的 `\unitlength` 值。



```
\begin{picture}(10.0,6.6)
\thicklines\put(0,0){\framebox(10.0,6.6){}}
\put(5.0,6.3){\makebox(0,0){\bfseries The Outer Picture}}
\thinlines
\put(.5,.5){\setlength{\unitlength}{1mm}}
\begin{picture}(50,25)
\put(0,0){\framebox(40,25){Sub Picture 1}}
\put(10,20){\circle*{1}} \put(10,20){\makebox(0,0)[l]{(10,20)}}
\put(4,4){\vector(-1,-1){4}}
\put(5,5){\makebox(0,0)[lb]{(0,0) Origin 1}} \end{picture}}
\put(5.5,0.5){ ... Sub Picture 2 ... }
```

```

\put(0.5,3.5){ ... Sub Picture 3 ... }
\put(5.5,3.5){ ... Sub Picture 4 ... }
\put(5,0.1){\makebox(0,0)[b]
    {\bfseries Bottom of Outer Picture}}
\end{picture}

```

最外层图形的单位长度设为  $UL=1\text{ cm}$ ，该图形宽  $10\text{ cm}$ ，高  $6.6\text{ cm}$ 。它的图形要素中有一个粗线的方框，这个方框的尺寸与图形的尺寸一样，其它的元素有：两块文本 ‘The Outer Picture’ 和 ‘The Bottom of the Outer Picture’，四幅小图形，每一幅的取  $UL=1\text{ mm}$  的单位长度和  $40\times 25\text{ mm}$  的尺寸。在每个小图形中的对象相对于它们自己的原点定位。在每个小图形中  $\bullet$  符号都具有相同的坐标  $(10,20)$ 。

通过使用嵌套图形，可以简化逻辑上彼此相关的特定对象之间的相互定位，从而减少了定位时可能出现的错误。当调用 `\put` 命令时把 `\unitlength` 命令设成不同值，尤其如此。

### §6.5.3 存贮部分图形

可以把图形中的一组图形元素用特定的名称保存成子图，然后就能随时调用整组命令，而不需一条一条地调用。

首先必须为每个子图起一个名称，所用命令为：

```
\newsavebox{\子图名称}
```

这样就会创建一个名叫 `\子图名称` 的用于保存图形的盒子。然后，用下面的命令保存子图：

```
\savebox{\子图名称}(x-尺寸, y-尺寸)[位置]{子图}
```

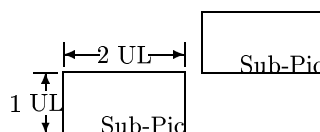
这里的参数值  $(x\text{-尺寸}, y\text{-尺寸})$  与位置同 6.4.2 节 `\makebox` 中的含义一样。

如果图形命令 `\子图` 只是一小块文本，那么这条命令就与 `\makebox` 命令几乎完全一样，除了文本不是显示出来，而是存贮到 `\子图名称` 中。子图可以用下面这条命令当做一个图形元素安置在主图内的任何地方。

```

\usebox{\子图名称}
\newsavebox{\sub}
\savebox{\sub}(2,1)[br]{\small Sub-Pic}
....
\put(0.7,0.0){\framebox{\usebox{\sub}}}
\put(3.0,1.0){\framebox{\usebox{\sub}}}

```



这个例子看不起并不怎样，因为这里的 `\savebox` 和 `\usebox` 命令如果用 `\framebox` 结合 `\multiput` 命令取代，简单地就可以得到同样的结果。然而，这两条命令的主要优势并不是安置多重文本，而是更复杂的子图组合。

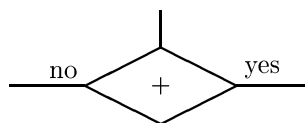
这里要指出，`\savebox` 命令可以在 `picture` 环境外面调用，甚至可以把它放在导言中。这样该子图就可以在整篇文档的所有 `picture` 环境中使用。

然而, 如果 `\savebox` 是定义在一个环境中, 那么其中的值当环境结束时也就没有了。

放在 `\savebox` 中的图形元素将会根据盒子被构造时候的单位长度确定尺寸。但不会受随后 `\unitlength` 改变的影响。

```
\newsavebox{\testcon}
\savebox{\testcon}(0,0){%
  \thicklines
  \put(0,0.5){\line(-2,-1){1.0}}
  \put(0,0.5){\line(2,-1){1.0}}
  \put(0,-0.5){\line(-2,1){1.0}}
  \put(0,-0.5){\line(2,1){1.0}}
  \put(0,0.5){\makebox(0,0)[b]
    {\put(0,0){\line(0,1){0.5}}}}
  \put(1.0,0){\line(1,0){1.0}}
  \put(-1.0,0){\line(-1,0){1.0}}
  \put(-1.1,0.1){\makebox(0,0)[br]{no}}
  \put(1.1,0.1){\makebox(0,0)[bl]{yes}}}
```

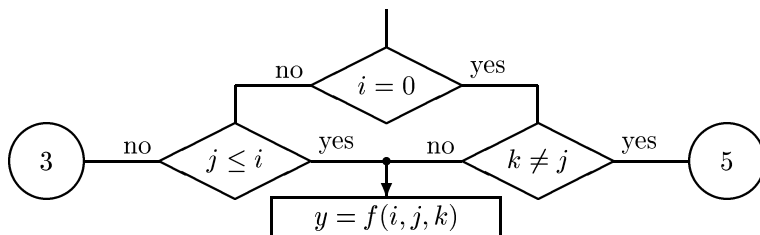
这定义了条件分枝符号, 它经常用在计算机程序的流程图中。



+ 表示符号 `\textcon` 的参考点。

在上面的 `\savebox` 命令中, 参数值 (0,0) 定义了一个零宽度和零高度的盒子, 这样它的中心就放在 `\put` 命令中安置坐标定义的地方。因为我们想要菱形的中心就是这个参考点, 因此菱形上面的竖直线必须没有高度, 否则它的出现就会使得整幅图的中心偏离菱形的中心。因此竖线 (`\line(0,1){0.5}`) 就放在一个零尺寸的 `\makebox` 盒子中。这样整幅图的参考点 (如外面的 `\put` 命令中一致) 就是 + 号所表示的地方。

现在 `\testcon` 命令可以很容易地与其它图形要素组合起来。



```
\begin{picture}(10,3) \thicklines
\put(5,2){\usebox{\testcon}\makebox(0,0){\$i=0\$}}
\put(3,1){\usebox{\testcon}\makebox(0,0){\$j\le i\$}}
\put(7,1){\usebox{\testcon}\makebox(0,0){\$k\neq j\$}}
\put(0.5,1){\circle{1.0}\makebox(0,0){3}}
\put(9.5,1){\circle{1.0}\makebox(0,0){5}}
\put(5,1){\vector(0,-1){0.5}\circle*{0.1}}
```



```
\put(3.5,0){\framebox(3,0.5){$y = f(i,j,k)$}}
\end{picture}
```

这个例子说明，在一条 `\put` 命令中可以包含不只一条图形对象。这里把表示符号的 `\usebox{\testcon}` 命令同包含文本的 `\makebox(0,0)` 放在了一起。类似地，`\circle{1.0}` 命令分别同其中的居中数字 3 和 5 放在一条 `\put` 命令中。最后，`\vector` 和 `\circle` 命令也放在了一起。当多条命令放在同一 `\put` 命令中时，在图形元素之间必须不能有空格。

也可以用 `\savebox` 命令存贮一个完整的 `picture` 环境。这时参数值 (x-尺寸,y-尺寸) 和位置是被忽略的，因为在 `picture` 环境中就有这些尺寸定义。语法为

```
\savebox{\图形名称}{\begin{picture}(x-尺寸, y-尺寸)
... ..
\end{picture} }
```

如果要把这样存贮的图形做为子图来使用，(x-尺寸,y-尺寸) 应预设为 (0,0)，因为这样使得可以相对于子图定位。被保存的图形将会根据外面的 `picture` 环境进行放缩。前面的示例符号 `\testcon` 可以这样保存起来：

```
\savebox{\testcon}{\begin{picture}(0,0) ...
\end{picture} }
```

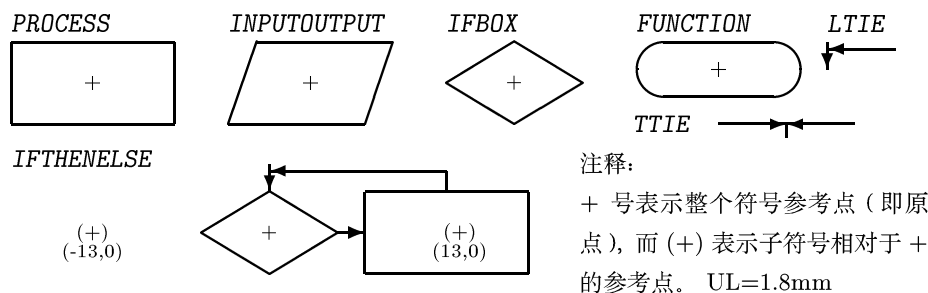
这里的 ... 代表前一页中定义 `\testcon` 时相同命令集合，从 `\thicklines` 开始，到最后一个 `}` 结束。

把子图作为 `picture` 环境保存起来的另一个好处是：即使点 (0,0) 并不是符号的真正中心，但也可以把它设成外部 `\put` 命令的参考位置点。这也就是说在 `\testcon` 的定义中，我们可以把

```
\put(0,0.5){\makebox(0,0)[b]{\put(0,0){\line(0,1){0.5}}}}
```

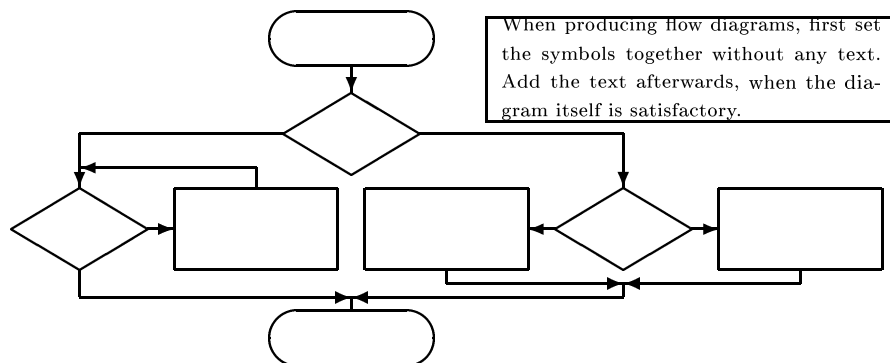
改成 `\put(0,0.5){\line(0,1){0.5}}`。也就是现在不需要‘隐藏’会把符号中心从点 (0,0) 移开的竖直线。

**练习 6.8:** 下面这些符号经常出现在计算机程序的流程图中。生成给定的盒子，并把它保存为具有相应名称的符号。用这些较简单的符号组合成复杂的符号。注意连接时的相对坐标和参考点。





利用上面的符号生成下面这幅图。



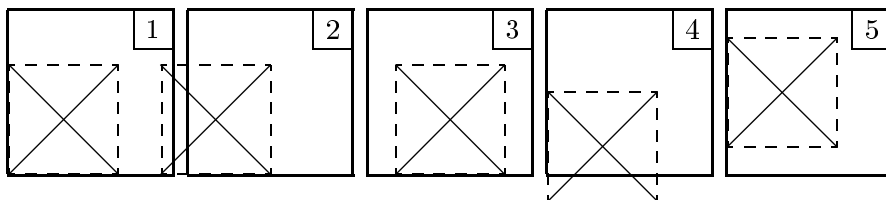
#### §6.5.4 图形环境的广义语法

在 `picture` 环境的广义语法中多了一对坐标，其为可省参数值：

`\begin{picture}(x-尺寸, y-尺寸)(x-偏移, y-偏移)`

图形命令 `\end{picture}`

在这种形式中， $(x\text{-偏移}, y\text{-偏移})$  定义了左下角的坐标。这就是说对环境中的所有 `\put` 命令，要从其定位坐标中减去  $x\text{-偏移}$  和  $y\text{-偏移}$ ，这样整幅图就向左平移了  $x\text{-偏移}$  单位，向下平移了  $y\text{-偏移}$  单位。



这里我们给出了五张图形，每个图形中有一个细线方框，其宽  $3UL$ ，高  $3UL$  ( $UL=7.2mm$ )。在每个方框中，用 `\put` 加入了一个子图，其中有  $2 \times 2UL$  的虚线框及两条对角线。每幅图除了偏移外都是一样的。

1. `\put(0,0){\begin{picture}(2,2)(0,0)`      子图 `\end{picture}}`
2. `\put(0,0){\begin{picture}(2,2)(0.5,0)`      子图 `\end{picture}}`
3. `\put(0,0){\begin{picture}(2,2)(-0.5,0)`      子图 `\end{picture}}`
4. `\put(0,0){\begin{picture}(2,2)(0,0.5)`      子图 `\end{picture}}`
5. `\put(0,0){\begin{picture}(2,2)(0,-0.5)`      子图 `\end{picture}}`

在情形 2 和 4 中，部分子图位于主图边界外面，当有偏移时必须考虑这一点。

## §6.5.5 更多的例子

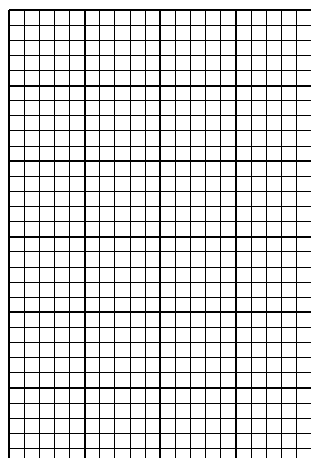
到现在为止，我们已经通过举例，相当详细地说明了各种图形元素。然而，我们有点儿冷落了 `\multiput` 命令，因此这里给出一些例子说明它的用法。（在 6.3 节中描述了 `\multiput` 命令。）

```
\multiput(0,0)(1,2){7}{\circle*{1}}
\multiput(8,0)(2,0){10}{\begin{picture}(0,0)
  \multiput(0,0)(1,2){7}{\circle*{1}}
\end{picture}}
```



第一条 `\multiput` 命令生成七个直径为 1mm 的点，它们从 (0,0) 开始，然后每次向右平移 1mm，同时向上平移 2mm。第二条 `\multiput` 命令生成 10 个子图，每个子图相对于前者向右平移 2mm。而子图本身就是前面第一条 `\multiput` 命令生成的七点。

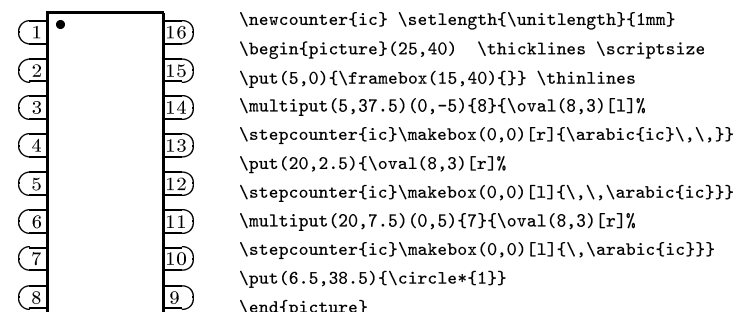
## 例：网格



```
\setlength{\unitlength}{0.1in}
\begin{picture}(20,30)
\linethickness{0.25mm}
  \multiput(0,0)(10,0){3}{\line(0,1){30}}
  \multiput(0,0)(0,10){4}{\line(1,0){20}}
\linethickness{0.15mm}
  \multiput(5,0)(10,0){2}{\line(0,1){30}}
  \multiput(0,5)(0,10){3}{\line(1,0){20}}
\linethickness{0.075mm}
  \multiput(1,0)(1,0){19}{\line(0,1){30}}
  \multiput(0,1)(0,1){29}{\line(1,0){20}}
\end{picture}
```

首先画出三条粗 0.25mm，长 30UL 的竖线，同样粗细的四条长 20UL 的横线，它们都是从 (0,0) 开始，彼此间距 10UL。而另两条竖线和三条横线粗 0.15mm，是从 (5,0) 和 (0,5) 开始画。最后我们画出了粗 0.075mm 的 19 条竖线和 29 条横线，它们彼此间隔 1UL。

## IC 符号



```
\newcounter{ic} \setlength{\unitlength}{1mm}
\begin{picture}(25,40) \thicklines \scriptsize
\put(5,0){\framebox(15,40){}} \thinlines
\multiput(5,37.5)(0,-5){8}{\oval(8,3)[l]}%
\stepcounter{ic}\makebox(0,0)[r]{\arabic{ic}\,,}%
\put(20,2.5){\oval(8,3)[r]}%
\stepcounter{ic}\makebox(0,0)[l]{\,,\,\arabic{ic}}}%
\multiput(20,7.5)(0,5){7}{\oval(8,3)[r]}%
\stepcounter{ic}\makebox(0,0)[l]{\,,\,\arabic{ic}}}%
\put(6.5,38.5){\circle*{1}}
\end{picture}
```

这个例子中的两条 `\multipt` 命令都包含两个图形对象：半个卵形线和自动增加的文本。第一条 `\multipt` 命令从上到下生成左边的对象，而第二条是从下向上生成右边的对象。命令 `\newcounter{ic}` 建立一个叫 `ic` 的新计数器，然后用 `\stepcounter{ic}` 逐次增 1，并用 `\arabic{ic}` 来以阿拉伯数字显示出它的值。（在上面的示范源文本中，每条 `\multipt` 命令都占有两行；而事实上，这条命令的输入必须没有空格或回车以避免图形的水平移位！）

**练习 6.9:** 利用右面的厘米尺为模型，生成 10cm 的横尺和竖尺。



**练习 6.10:** 改进练习 6.5 中的方格纸，方法是如上面例子那样把每个第 5 条和第 10 条直线变粗。并且沿外边界标出每个第 10 条直线。

### §6.5.6 扩展软件包 2ε

在标准 L<sup>A</sup>T<sub>E</sub>X 2<sub>ε</sub> 安装中，为了增强 `picture` 环境的功能，提供了两个软件包。

#### 去掉某些限制

利用打印机驱动程序的一些功能，可以去掉对斜线长度和倾角的限制。软件包 `pict2e` 就是为了激活特定驱动程序的这种能力而设计的。结果可能不会是真的与设备无关，但应该对于支持这种绘图操作的驱动程序都行得通。

利用这个软件包，圆也可以具有任意直径，而不只是局限于那些可用的特殊圆字体。

像通常那样，把这个软件包用 `\usepackage{pict2e}` 包含在导言中。

注意：在编写本书时 (August 15, 1999) 这个软件包还没有付诸于实用，因此这里无法给出其选项或驱动程序选择。

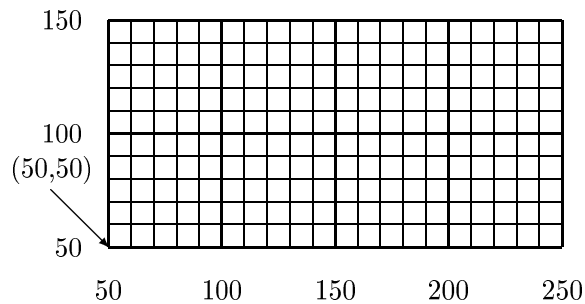
#### 格纸

软件包 `graphpap` 增加了一条绘制格纸的命令

2ε `\graphpaper[数](x,y)(lx,ly)`

这条命令把方格的左下角放在  $(x,y)$  处，它有  $lx$  单位宽， $ly$  单位高。每隔给定的数单位就划一条格线，而每个第 5 条格纸要粗一些，并且加上标记。如果没有给出数，那就假定它是 10。所有参数值必须是整数，而不能是小数。

例如，`\graphpaper(50,50)(200,100)` 的结果为 (UL=0.3mm):



### §6.5.7 一般性建议

在实际应用中，每个用户都会形成自己使用 `picture` 环境的风格。这里我们要讲述由 Leslie Lamport 给出的一些建议，我们对这些建议是非常赞成的，而且也增加了一些我们自己的看法。

1. 对于绘制和定位单个图形元素，各种网格尺寸的格纸是相当有用的。借助于这种网格，或者 `\graphpaper` 命令，用户可以生成任意尺寸的格纸，然后复制足够份，以备将来使用。
2. 网格的最小间距应就取为单位长度的大小。这样可以避免定位时出现小数。
3. 如果图形中没有斜线，那么图形元素的位置坐标可以直接从格纸上读出来。单个对象的参考点应是一个格点，并不是位于格点之间。
4. 对于斜线，只有有限的几个斜率（6.4.3 节）。当指定了一个可以接受的斜率后，那么直线的端点应该放在格点上。对箭头也要这样做（6.4.4 节）。（如果包含了 `pict2e` 软件包就不必这样做了。）
5. 图形应该利用子图构成，或者是子图的子图构成（6.5.2 节）。这样就可以通过相对于子图定位，从而简化图形绘制中的这种操作。
6. 经常用的子图应该保存在 `\savebox` 中，并收集到一个或多个独立的文件中。利用这种方法，经过几年的积累，我们可以建立自己的符号库。建立这些符号的文档时应该包含它们的名称、参考点和联系点，这样其它用户也可以用它们了。

在 `picture` 环境一个很小的错误，也有可能導致毁灭性的后果。因此如果得到的结果完全出乎意料之外，也没有必要焦虑。例如，若一幅图形设计时是按照单位长度为 `1cm` 进行的，而没有设定 `\unitlength`，这样 `LaTeX` 就会取默认值 `1pt`，从而使得图形几乎缩成一点，其中包含的文本不可想像地都显示在一点上。如果图形显示在主图区域外面，对于这种出乎意料的结果是由于不正确的定位造成的，可能是错误的符号或者漏掉了小数点。

有时候在水平定位中，尤其是几个图形用一条 `\put` 或 `\multiput` 命令安置时，会出现一些小错。这可能是在图形命令之间加了空格。这样的空格

也会被当做图形元素，从而把下一个符号从右进行了一点移位。因此在元素命令之间应该没有空格或回车（新行）。如果还是有错误，那就试着换一下安置命令中的元素顺序。如果这样做还行不通，那就只好每个元素用一条 `\put` 或 `\multiput` 命令了。

## §6.6 浮动表格和插图

`picture` 环境生成的图形显示在输入文本的地方，前后都是其它文本。如果当前页上有足够的地方，这通常没有任何问题，而且也是我们所期望的。然而如果插图（或者表格）很高，在当前页上放不下来，那么就会结束当前页，把插图（或表格）放在下一页的顶部。这样当前页的格式就会很难看。

如果存在下面这种机制就相当好了：若当前页空间足够，就把插图或表格放在当前页上，否则就留到下一页上考虑，直到找到适当的地方放置。由于表格或插图通常带有标题或说明，这些东西也应随同移动。

在 4.8.4 节中已引进了相应于表格的这种机制。下面将给出浮动的完整描述。

### §6.6.1 浮动安置

L<sup>A</sup>T<sub>E</sub>X 确实确实地按照上面所要求的那样，做到了插图和表格连同标题和说明在一起的浮动。这种机制是用下面命令调用的：

```
\begin{figure}[位置] 插图 \end{figure}
\begin{figure*}[位置] 插图 \end{figure*}
\begin{table}[位置]   表格 \end{table}
\begin{table*}[位置]  表格 \end{table*}
```

\*- 形式只适用于两列页面格式，它使得插图或表格占据两列，而不是正常情形的一列。当页面格式是单列时，它的作用同标准形式一样。

在上面的语法中，插图和表格就是要浮动的内容，它是包含在 `picture` 或 `tabular` 环境中的，还有可能出现的 `\caption` 命令（我们稍后在 6.6.4 节中介绍这一命令）。

而参数值 `位置` 定义了插图或表格允许出现的地方。`位置` 由零到四个字母组成，因此取值有很多可能，字母的意义如下：

- h** here: 浮动对象可以位于环境输入时所处的地方；它不能用于 \*- 形式；
- t** top: 浮动对象可以出现在当前页的顶部，条件是要有足够的空间以容纳它自己和其前面的文本；如果这行不通，就会把它加到下一页的顶部；页后续文本仍然显示在当前页上，直到正常地分页；（对于两列格式，上面叙述中的页换成列就可以了）；

- b bottom: 浮动对象可以出现在当前页的底部; 后续文本继续到在当前页上有放下浮动对象的足够地方为止; 如果在当前页上已经没有足够地方, 浮动对象会被放到下一页的底部; 在 \*- 形式中不能取这个值;
- p page: 浮动对象可以放在一个特殊页 (或列) 上, 该页 (或列) 上只有插图和 / 或表格;
- 2ε ! 与其它字母的组合一起使用, 它会去掉在 6.6.3 节中描述的关于间距和数值限制。

参数值可以组合, 形成几种可能。如果没有给定任何值, L<sup>A</sup>T<sub>E</sub>X 假定它是标准组合 `tbp`。

安置参数值使得定位浮动对象有几种可能, 但是实际的插入点是符合下述规则的最早可能点:

- 在它定义的前面一页上再没有浮动对象;
- 插图和表格是按照在文本中定义的先后顺序输出的, 因此不会有浮动对象会在前面定义的同类型对象之前输出; 然而插图和表格输出顺序有可能混杂在一起; 在两列格式中的双列 \*- 浮动对象也有可能不符合这一顺序;
- 浮动对象可会出现在安置参数值 位置 所允许的位置上; 若没有该参数值, 就会用标准组合 `tbp`;
- 除非在 位置 中包含了 !, 定位要遵从在 6.6.3 节中描述的样式参数的限制;
- 对于组合 `ht`, 优先考虑 `h`; 即使当前页顶部有足够的空间, 浮动对象也会被插入在定义点。

当使用了 `\clearpage`, `\cleardoublepage` 或者 `\end{document}` 命令时, 所有还没输出的浮动对象就会显示在单独一页或列上, 而不会考虑它的定位参数。

### §6.6.2 延迟浮动

有的时候可能不希望浮动对象出现在某些页面上, 例如不要出现在标题页的顶部。(L<sup>A</sup>T<sub>E</sub>X 会自动地校正这种情形。) 然而, 也有时候需要暂时抑制浮动。我们可能希望它位于页面顶部, 但是我们不希望它位于一节引用它的那节的前面。命令

2ε `\suppressfloats[位置]`

确保在当前页的 位置 定义的地方没有其它浮动对象。如果没有可省参数 位置, 就会抑制所有的浮动; 否则 位置 可以是 `t` 或 `b`, 但不会全有。

注意 `\suppressfloats` 并没有抑制当前页所有的浮动, 它只是抑制位于从调用该命令开始到该页结束之间的浮动。因此来自于前一节的浮动仍旧有可能出现在当前页上。

`\suppressfloats` 命令和 `!` 位置参数都是在  $\text{\LaTeX 2}_\epsilon$  中才有的，它们的目的就在于尝试给作者更大的控制浮动安置中可能出现的反复无常行为。

### §6.6.3 浮动中的样式参数

有很多影响浮动安置的样式参数，用户可以修改它们：

`topnumber` 可以位于一页顶部的最多浮动对象数。

`bottomnumber` 可以位于一页底部的最多浮动对象数。

`totalnumber` 不考虑位置，可以位于一页中最多的浮动对象数。

`dbltopnumber` 同 `tatalnumber` 一样，只是表限定的是双列格式中横跨两列的浮动对象数。

上述参数都是计数器，可以用命令 `\setcounter{计数器}{数}` 来给它设定新值，这里 `计数器` 指的是是计数器的名称，而 `数` 是将要设定的新值。

`\topfraction` 是一个小数，定义页面顶部多大部分可以放浮动对象。

`\bottomfraction` 是一个小数，定义页面底部多大部分可以放浮动对象。

`\textfraction` 该数规定了页面多大部分必须填充以文本。这表示一个最低限度，因此无论顶部，还是底部，总共放浮动对象的部分不能超过  $1 - \text{\textfraction}$ 。

`\floatpagefraction` 在开始新页之前浮动页中填充的浮动对象所占部分的最低限度。

`\dbltopfraction` 同 `\topfraction` 一样，只是它对应于两列页面格式中双列浮动对象。

`\dblfloatpagefraction` 同 `\floatpagefraction` 一样，只是它对应的是两列页面格式中双列浮动对象。

要用 `\renewcommand{命令}{小数}` 来改变这些样式参数的值，这里 `命令` 表示参数名，`小数` 是新数，每个都必须小于 1。

`\floatsep` 出现在页面顶部或底部中的浮动对象之间的竖直距离。

`\textfloatsep` 页面顶部和底部中的浮动对象与文本之间的竖直距离。

`\intextsep` 当用 `h` 安置参数值时出现在一页中间的浮动对象与上下正文之间的竖直距离。

`\dblfloatsep` 与 `\floatsep` 一样，只是它对应于两列页面格式中双列浮动对象。

`\dbltextfloatsep` 同 `\textfloatsep` 一样，只是它对应于两列页面格式中的双列浮动对象。

这一组样式参数都是橡皮长度，可以用 `\setlength` 命令修改它们的值（2.4.2 节）。

$\boxed{2\epsilon}$  `\topfigrule` 在一页顶部浮动对象之后被执行的命令。可以用它加上标尺以把浮动对象与正文分开。但是这里加入的标尺必须是零高度。



<sup>2ε</sup> `\botfigrule` 类似于 `\topfigrule`，但它是在位于页面底部浮动对象之前被执行。

<sup>2ε</sup> `\dblfigrule` 类似于 `\topfigrule`，但它是相应于双列浮动对象。

这三条命令通常什么都不做，但必要时可以重定义它们。例如，可以用下面的输入在顶部浮动之后加上粗 0.4pt 的标尺：

```
\renewcommand{\topfigrule}{\vspace*{-3pt}
\rule{\columnwidth}{0.4pt}\vspace{2.6pt} }
```

由于在 `\vspace*` 中是负的参数值，整个竖直距离还是零，这满足命令的要求。

所有单列样式参数在两列页面格式中也有作用，但它们只适用于填充一列的浮动对象。

#### §6.6.4 浮动对象中的说明

可以用下面的命令生成插图的说明或表格的标题：

```
\caption[短标题]{标题文本}
```

要把这条命令放在 `figure` 或 `table` 环境中。标题文本 就是与浮动对象显示在一起的文本，它可以相当长。短标题 可以省略的，它是出现在插图或表格列表中的文本（3.4.4 节）。如果不给出这段文本，那它就等于 标题文本。如果 标题文本 长度超过 300 个字符，或者比一行还长的话，那就应该给出 短标题。

在 `table` 环境中，`\caption` 命令生成形如 ‘Table *n*: 标题文本’ 的标题，而在 `figure` 环境中，则是形如 ‘Figure *n*: 标题文本’ 的说明，这里的 *n* 是自动给出的顺序编号。在文档类 `article` 中，插图和表格的编号是从 1 开始，直到文档结束。而对于 `report` 和 `book` 类，每章单独编号，形式为 *c.n*，这里 *c* 表示当前章号，*n* 是顺序号，每章开始被重设为 1。插图和表格的编号是相互独立的。

如果不要编号的话，可以不用 `\caption` 命令，因此包含在浮动环境中的任意文本都会随着其它内容一起移动。而 `\caption` 比普通文本强的地方就在于自动编号，而且其会自动列在插图和表格列表中。然而，也可以按 3.4.4 节所讲的那样用下面的命令手工向列表中增加一些项：

```
\addcontentsline 和 \addtocontents
```

当 `\caption` 命令位于浮动环境中其它材料的前面，那就可以形成一个标题，即编号和文本显示在表格或插图的上方。如果该命令出现在所有其它浮动命令的后面，那么就会在对象的下方加上说明。也就是说，`\caption` 只是浮动中的一项，其中文本出现在顶部（标题）或底部（说明）与用户放的位置有关。

如果 标题文本 的长度不足一行，那它就会居中排列，否则它同正常段落

表 6.1: Computer Center Budget for 1995

Nr.	Item	51505	52201	53998	Total
1.1	Maintenance	130 000		15 000	145 000
1.2	Network costs	5 000		23 000	28 000
1.3	Repairs	25 000	6 000		31 000
1.4	Expendables		68 000		68 000
1.	Total	160 000	74 000	38 000	272 000

一样。可以通过把命令放在子段盒子或者小页中来相应于表格或插图调整总宽度。例如,

```
\parbox{ 宽度 }{\caption{ 标题文本 }}
```

下面的例子说明了如何把文本、表格和图形命令组合进浮动对象。

### §6.6.5 浮动示例

前两个表格是用下述文本生成的:

```
\begin{table} \caption{Computer Center Budget for 1995}
\begin{tabular}{|l|l|l|r|r|r|} ... \end{tabular}
\end{table}
```

(这段文本是输入在上一节最后一段的前面。下面的第二个表格就输入在此处。)

```
\begin{table}
\caption{\textbf{Estimates for 1996} {\em A continuation...}}
\begin{tabular}{|l|l|l|r|r|r|} .. ... \end{tabular}
\end{table}
```

由于在 `table` 环境中没有给出安置参数值, 因此这里用的是标准值 `tbp`。第一个表格被放在该页的顶部, 因为它输入的很早 (在上一节中), 而且那里有足够的空间放下它。那么命令 `\suppressfloats` (6.6.2 节) 被调用, 以禁止在这一页上再出现其它的浮动对象。因此第二个表格就浮动到下一页的顶部。

窄的插图或表格可以并排放在一起, 如下例所示 (结果显示在下一页的底部)。

```
\begin{figure}[b]
\setlength{\unitlength}{1cm}
\begin{minipage}[t]{5.0cm}
\begin{picture}(5.0,2.5) ... \end{picture}\par
\caption{Left}
```

**表 6.2:** *Estimates for 1996 A continuation of the previous budget is no longer practical since, with the installation of the new computing system in 1995, the operating conditions have been completely overhauled.*

Nr.	Item	51505	52201	53998	Total
1.1	Maintenance	240 000			240 000
1.2	Line costs	12 000	8 000	36 000	56 000
1.3	Training			50 000	50 000
1.4	Expansion	80 000	3 000		83 000
1.5	Expendables		42 000		42 000
1.	Total	332 000	53 000	86 000	471 000

```

\end{minipage}
\hfill
\begin{minipage}[t]{6.0cm}
\begin{picture}(6.0,3.0) ... \end{picture}\par
\caption{Right}
\end{minipage}
\end{figure}

```

这两幅图连同其说明分别放在宽 5cm 和 6cmminipage 环境中。两个小页之间用 \hfill 间距分开。定位参数值 t 使得小页环境的第一行是对齐的 (4.7.3 节)。在 figure 环境中的所有结构当做一个整体进行浮动。

那么现在就有问题了, 既然两幅插图的高度不等, 这里要求它们的顶行对齐, 那为什么是底边在同一水平线上呢? 答案是 picture 环境建立起一个 LR 盒子 (4.7.1 节), 它包含所有的图形命令, L<sup>A</sup>T<sub>E</sub>X 认为这些命令就是一行输入, 基线就是图形的底边。在这里的两个小页环境中, picture 环境是其中的第一项, 因此也就是文本的第一个逻辑行。从而它们的基线就用来进行小页的竖直对齐。

如果要把两个图形沿顶边对齐, 那就需要在每个小页环境中 picture 环

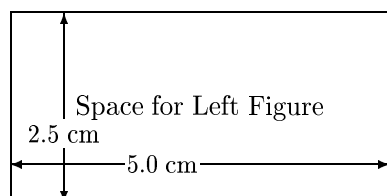


图 6.1: Left

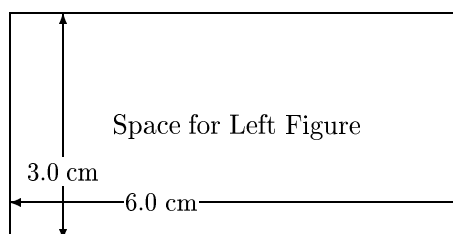


图 6.2: Right

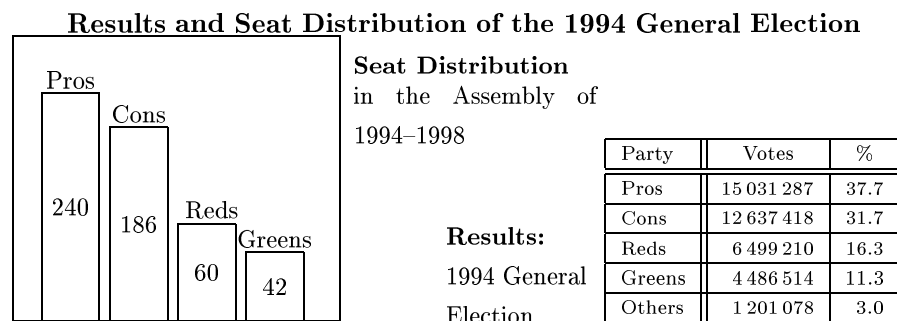
境之前插入一个没有内容的第一行（4.7.4 节）。例如这可以是 `\mbox{}`。

在浮动对象中应用盒子命令是可以完全自由定位的。如果说明文本位于表格或插图的一旁，而不是上面或下面，那么对象可以放在小页或子段盒子中，选择适当的对齐参数值。下面给出一个例子：

```
\begin{table}[b]
  \centerline{\bfseries Results and Seat Distribution of the ...}
  \mbox{\small
    \begin{minipage}[b]{7.7cm}
      \begin{minipage}[t]{4.4cm}
        \mbox{}\ \ \ \unitlength0.75cm
        \begin{picture}(5.75,5.0) ... \end{picture}
      \end{minipage} \hfill
      \parbox[t]{3.2cm}{\makebox[0cm]{\ \ \ \bfseries Seat ...} ...}
    \ \ \mbox{}
  \end{minipage}
  \hspace{-3cm}
  \begin{minipage}[b]{7cm}
    \parbox[b]{2.5cm}{\bfseries Results:}\ \ \ General Election...
    \hfill
    \begin{tabular}[b]{|l||r|r|} ... \end{tabular}
  \end{minipage} }
\end{table}
```

在这里需要使用嵌套竖直盒子。最左边是由图形和右上角的标题组成，它位于一个宽 7.7cm 的小页中，其中的图形所在的另一个小页宽 4.4cm，而文本又位于一个宽 3.2 cm 的子段盒子中。这两者是顶行对齐。为此需要在 `picture` 前面加上一个没有内容的行 `\mbox{}`（4.7.4 节），以使得顶行与子段盒子对齐。

右边是由一个宽 7cm 的小页组成，它向左平移了 3cm，并与左面的小页



沿底边对齐。它由一个放文本宽 2.5cm 的子段盒子和一个表格组成，表格自动就是一个竖直盒子。这两者是底部对齐的。

### §6.6.6 在正文中对插图和表格的引用

表格和插图的自动编号，就意味着在写作时作者并不知道它们的编号。而他(或她)又希望能够用这些对象的编号引用它们，如‘Figure 3’或‘Table 5 illustrates’，因此需要找到一种引用的方法。而仅仅跟踪已调用的 `\caption` 中的编号又是不够的，因为文档可能不是按顺序写作的，修改时又可能插入或删除插图或表格。

这个问题可以借助于 L<sup>A</sup>T<sub>E</sub>X 的交叉索引系统来解决，在 8.3.1 节中将详细讲解。基本命令是

```
\label{ 名称 }      \ref{ 名称 }
```

这里给正文中要用到的插图或表格编号赋予一个关键词 `名称`。关键词可以是字母、数字或符号的任意组合。赋值是用 `\label` 命令放在 `\caption` 命令的说明文本的任何地方就可以了；在正文中，命令 `\ref` 就会插入相应关键词所对应的编号。

下面用一个例子就可以很好地说明这一操作。在 159 页上的表格实际上写为

```
\caption{\label{budget95} Computer Center ... }
```

因此在正文中用 `Table \ref{budget95}` 可以生成‘Table 6.1’。

还有另一条引用命令 `\pageref`，它可以生成被引用对象所在那一页的页码。例如，刚刚几行前，就是用‘在 `\pageref{budget95}` 页上’得到文本‘在 159 页上’。