Tikz 学习笔记

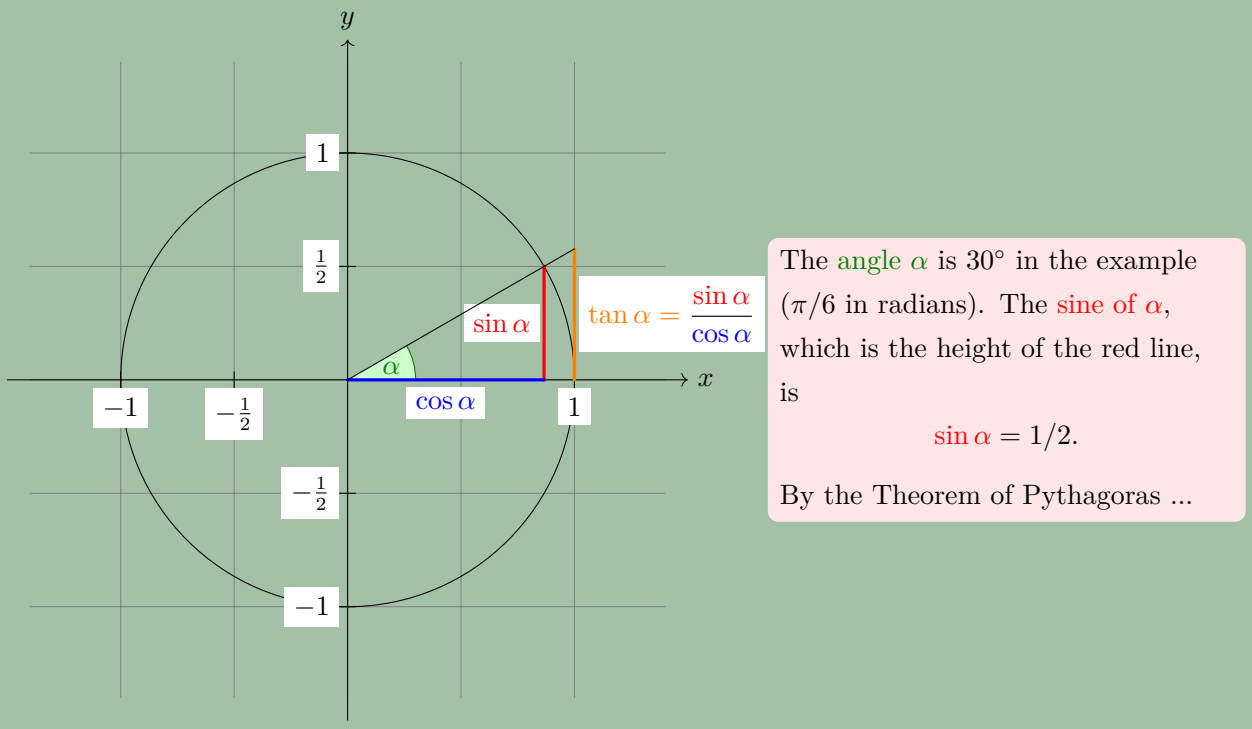
2017 年 4 月 29 日

目录

1 三类例子

1.1 单位圆与三角函数线

本小节展示的代码绘制下面的图形




```
1  \begin{tikzpicture}
2  [scale=3,line cap=round,
3  % Styles
4  axes/.style=,
5  important line/.style={very thick},
6  information text/.style={rounded corners,
7                           fill=red!10,inner sep=1ex}]
8  % Colors
9  \colorlet{anglecolor}{green!50!black}
10 \colorlet{sincolor}{red}
11 \colorlet{tancolor}{orange!80!black}
12 \colorlet{coscolor}{blue}
```

1 行，开启 tikzpicture 环境。

2 行，scale=3，缩放比例为 3。line cap=round，线的端头采用圆形（rect 为方形，butt 为无头）。

4 行，axes 采用默认样式。

5 行，定义 important line 的样式，只要求线宽为 very thick。

6 行，定义 information text 的样式，圆角边框，颜色为 red!10，文字与文字背景边界距离为 1ex。

9 行，命令 \colorlet 定义颜色 green!50!black 的名称为 anglecolor，可以用此名称引用该颜色。10、11、12 行类似。

```
13 % The graphic
14 \draw[help lines,step=0.5cm] (-1.4,-1.4) grid (1.4,1.4);
15 \draw (0,0) circle [radius=1cm];
16 \begin{scope}[axes]
17 \draw[->] (-1.5,0) -- (1.5,0) node[right] {$x$}
18           coordinate(x axis);
19 \draw[->] (0,-1.5) -- (0,1.5) node[above] {$y$}
20           coordinate(y axis);
21 \foreach \x/\xtext in {-1, -.5/-\frac{1}{2}, 1}
22 \draw[xshift=\x cm] (0pt,1pt) -- (0pt,-1pt)
23                   node[below,fill=white] {$\xtext$};
24 \foreach \y/\ytext in {-1, -.5/-\frac{1}{2}, .5/\frac{1}{2}, 1}
25 \draw[yshift=\y cm] (1pt,0pt) -- (-1pt,0pt)
26                   node[left,fill=white] {$\ytext$};
27 \end{scope}
```

14 行，\draw 命令开始绘图，grid 是网格命令，采用 help lines 的样式，步长 step=0.5cm，以点 (-1.4,-1.4) 到 (1.4,1.4) 为对角线。

15 行，画圆，圆心为 (0,0)，半径为 radius=1cm。

16 行，开启 scope 环境，即在整个大图中再绘制一个局部图。

17 行，从 (-1.5,0) 到 (1.5,0) 画线段，选项 [->] 为线段末端添加箭头；点 (1.5,0) 后的 node 命令为该点添加标签，选项 [right] 指示标签在点的右侧，标签是 LaTeX 符号 x。

21 行，\foreach 命令指示对数组中的每个元素进行后面的操作。\x/\xtext 表示数组元素可以是\x 形式的或\x/\xtext 形式的；同一个元素位置上用符号/并列两种形式时，该元素可用两种形式之一引用。数组 {1,...,10} 是公差为 1 的等差数列，数组 {1,1.5,,...,10} 是公差为前两项之差 0.5 的等差数列。

22 行，是 21 行的\foreach 命令指向的操作。选项 [xshift=\x cm] 指示将后面绘制的线段沿着 x 轴平移 \x cm。

23 行，node 命令为 22 行的点 (0pt,-1pt) 加标签，标签用 $\xtext$ 形式。选项 [below,fill=white] 表示标签在点下方，标签背景为白色。

27 行，结束 scope 环境。

```
28 \filldraw[fill=green!20,draw=anglecolor] (0,0) -- (3mm,0pt)
29 arc [start angle=0, end angle=30, radius=3mm];
30 \draw (15:2mm) node[anglecolor] {$\alpha$};
31 \draw[important line,sincolor]
32      (30:1cm) -- node[left=1pt,fill=white] {$\sin \alpha$}
33      (30:1cm |- x axis);
34 \draw[important line,coscolor]
35      (30:1cm |- x axis) -- node[below=2pt,fill=white]
36      {$\cos \alpha$} (0,0);
37 \path [name path=upward line] (1,0) -- (1,1);
38 \path [name path=sloped line] (0,0) -- (30:1.5cm);
39 \draw [name intersections={of=upward line and sloped line, by=t}]
40      [very thick,orange] (1,0) -- node [right=1pt,fill=white]
41 {$\displaystyle \tan \alpha \color{black}=
42 \frac{{\color{red}\sin \alpha}}{\color{blue}\cos \alpha}$}(t);
43 \draw (0,0) -- (t);
```

28 行，\filldraw 是填充颜色命令\fill 和绘图命令\draw 的结合。fill=green!20 表示用颜色 green!20 填充，draw=anglecolor 表示用颜色 anglecolor（已在 9 行定义）绘图。

29 行，arc 命令绘制圆弧线，选项 start angle=0 表示起始角度是 0°，end angle=30 表示终止角度是 30°，radius=3mm 表示半径是 3mm，默认圆心是 (0,0)。

30 行，极坐标点 (15:2mm) 是 15°，2mm 的点。

31 行，选项 important line，sincolor 已在 5 行，10 行定义。

32 行，node 命令在连接符号--之后，表示标签加在路径中间，标签 $\sin \alpha$ 是 LaTeX 符号 $\sin\alpha$。

33 行，表示过点 (30:1cm) 的竖直线与 x 轴的交点。

35 行，node 命令的选项 below=2pt 表示标签在路径下方 2pt 处，fill=white 表示标签文字背景填充为白色。

37 行，\path 命令声明它后面的对象是路径，这里的路径是个线段，选项 name path=upward line 即命名路径为 upward line。

39 行，选项表示命名交点，把路径 upward line 与 sloped line 的交点记命名为 t。

41 行，LaTeX 命令 \displaystyle 定义显示公式样式，注意颜色命令 \color{black} 只对 42 行的分数线有作用。

```
44 \draw[xshift=1.85cm]
45      node[right,text width=6cm,information text]
46 {
```

```
  The {\color{anglecolor} angle $\alpha$} is $30^\circ$ in the
  example ($\pi/6$ in radians). The {\color{sincolor}sine of
  $\alpha$}, which is the height of the red line, is
  \[
    {\color{sincolor} \sin \alpha} = 1/2.
  \]
  By the Theorem of Pythagoras ...
};
\end{tikzpicture}
```

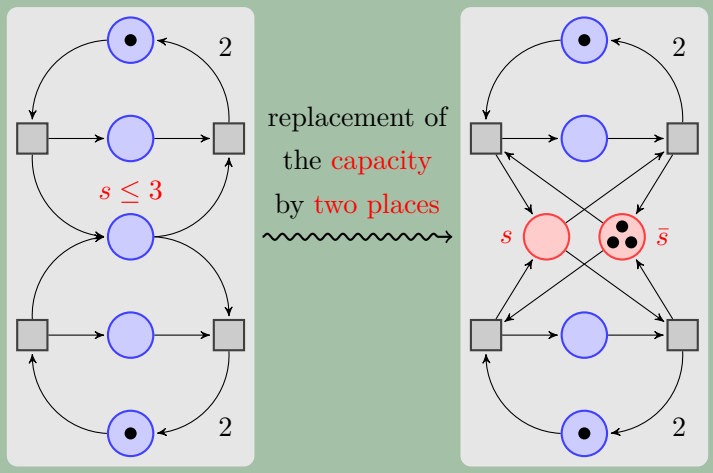
44 行，选项 [xshift=1.85cm] 将绘制的标签右移 1.85cm。

45 行至 54 行，给原点加的标签，按 44 行的选项平移。

55 行，结束 {tikzpicture} 环境。

1.2 A Petri-Net

本小节展示的代码绘制下面的图形




```
\begin{tikzpicture}
     [node distance=1.3cm,on grid,>=stealth',bend angle=45,auto,
      every place/.style= {minimum size=6mm,thick,draw=blue!75,fill=blue!20},
      every transition/.style={thick,draw=black!75,fill=black!20},
      red place/.style= {place,draw=red!75,fill=red!20},
      every label/.style= {red}]
```

2 行，node distance 确定 node 的间距，如下设置该间距

node distance=距离

当环境带有 on grid 选项时，该值指的是 node 的中心之间的距离；当环境无 on grid 选项时，该值指的是 node 边界之间的距离，初始值为 1cm。>=stealth’ 指示箭头样式。bend angle=45 指示路径

的转弯角度。auto 指示路径标签与路径的相对位置。用

auto=direction

设置 auto 的位置，其中 direction 可以是 left，right，false；当为路径标签选择 above，below 等位置时，或使用 anchor 选项时，auto 的值是 false；auto 的值默认是 scope 的设置。

3 行,every place 定义所有 place 的样式。minimum size=6mm 是最小尺寸为 6mm;draw=blue!75，用颜色 blue!75 画出边界线；fill=blue!20，用颜色 blue!20 填充标签。

5 行，定义 red place 样式，其中调用了 place 样式。

6 行，every label 定义所有 lable 的样式。

```
7  \node [place,tokens=1] (w1) {};
8  \node [place] (c1) [below=of w1] {};
9  \node [place] (s) [below=of c1,label=above:$s\le 3$] {};
10 \node [place] (c2) [below=of s] {};
11 \node [place,tokens=1] (w2) [below=of c2] {};
12 \node [transition] (e1) [left=of c1] {}
13                     edge [pre,bend left] (w1)
14                     edge [post,bend right] (s)
15                     edge [post] (c1);
16 \node [transition] (e2) [left=of c2] {}
17                     edge [pre,bend right] (w2)
18                     edge [post,bend left] (s)
19                     edge [post] (c2);
20 \node [transition] (l1) [right=of c1] {}
21                     edge [pre] (c1)
22                     edge [pre,bend left] (s)
23                     edge [post,bend right] node[swap] {2} (w1);
24 \node [transition] (l2) [right=of c2] {}
25                     edge [pre] (c2)
26                     edge [pre,bend right] (s)
27                     edge [post,bend left] node {2} (w2);
```

7 行,命令 \node 指示绘制标签。选项 place 已在 3 行定义。tokens=1 指示标记数目为 1,(w1) 中的 w1 是标签的名称。花括号无内容，表示标签无文字（但不能省略花括号）。

8 行，(c1) 指示标签的名称为 c1。选项 below=of w1 表示标签在 7 行定义的标签 w1 的下方。

9 行，选项 label=above:$s\le 3$ 指示在 node 的上面加 LaTeX 标签 $s \leq 3$。

12 行，选项 transition 已在 4 行定义。

13-15 行，从 12 行的 e1 分别向 w1，s，c1 画线，注意 13、14 行无分号，15 行有分号。（选项 pre，post）。bend left=angle，以出发点为原点，起点至终点的位移方向为初始边方向，逆时针方向为

正，这样规定下的 angle 是起点处的出发方向；负的 angle 是终点处的进入方向。bend right=angle 的意思与之类似，只是角的方向相反。如果不指定角度，它的默认值是前面最近出现的角度。

17-19 行，从 16 行的 e2 向 w2，s，c2 画线。注意 17、18 行无分号，19 行有分号。

23 行，node 给该行的 edge 加标签，标签内容是 2。选项 swap 指示标签的位置与 2 行的 auto 规定的位置相反。

27 行，node 给该行的 edge 加标签。标签内容是 2。

```
28 \begin{scope}[xshift=6cm]
29 \node [place,tokens=1] (w1' ) {};
30 \node [place] (c1' ) [below=of w1' ] {};
31 \node [red place] (s1' ) [below=of c1' ,xshift=-5mm]
32      [label=left:$s$] {};
33 \node [red place,tokens=3] (s2' ) [below=of c1' ,xshift=5mm]
34      [label=right:$\bar s$] {};
35 \node [place] (c2' ) [below=of s1' ,xshift=5mm] {};
36 \node [place,tokens=1] (w2' ) [below=of c2' ] {};
37 \node [transition] (e1' ) [left=of c1' ] {}
38                       edge [pre,bend left] (w1' )
39                       edge [post] (s1' )
40                       edge [pre] (s2' )
41                       edge [post] (c1' );
42 \node [transition] (e2' ) [left=of c2' ] {}
43                       edge [pre,bend right] (w2' )
44                       edge [post] (s1' )
45                       edge [pre] (s2' )
46                       edge [post] (c2' );
47 \node [transition] (l1' ) [right=of c1' ] {}
48                       edge [pre] (c1' )
49                       edge [pre] (s1' )
50                       edge [post] (s2' )
51                       edge [post,bend right] node[swap] {2} (w1' );
52 \node [transition] (l2' ) [right=of c2' ] {}
53                       edge [pre] (c2' )
54                       edge [pre] (s1' )
55                       edge [post] (s2' )
56                       edge [post,bend left] node {2} (w2' );
57 \end{scope}
```

28 行，开启 scope 环境，选项 xshift=6cm 使得环境生成的图右移 6cm。

31 行，选项 xshift=-5mm 使得标签左移 5mm。

32 行，给 node 加标签，选项 left 指示标签在 node 的左侧。

56 行，node 给 edge 加标签，标签内容是 2。标签位置参照 2 行 auto 的指示。

57 行，结束 scope 环境。

```
58 \begin{scope}[on background layer]
59 \node (r1) [fill=black!10,rounded corners,
60           fit=(w1)(w2)(e1)(e2)(l1)(l2)] {};
61 \node (r2) [fill=black!10,rounded corners,
62           fit=(w1' )(w2' )(e1' )(e2' )(l1' )(l2' )] {};
63 \end{scope}
64 \draw [shorten >=1mm,-to,thick,decorate,
65      decoration={snake,amplitude=.4mm,segment length=2mm,
66      pre=moveto,pre length=1mm,post length=2mm}]
67     (r1) -- (r2) node [above=1mm,midway,
68                        text width=3cm,align=center]
69                       {replacement of the \textcolor{red}{capacity}
70                        by \textcolor{red}{two places}};
71 \end{tikzpicture}
```

58 行，选项 on background layer 指示开启的 scope 环境所在的层属于主层（即 tikzpicture 环境所在层）之下的层。

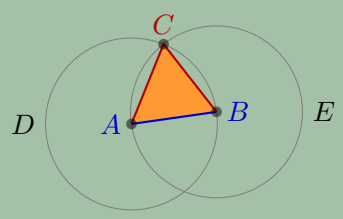
59 行，选项 rounded corners 指示标签边框用圆角。

60 行，选项 fit 生成一个盒子（fit=(w1)(w2)(e1)(e2)(l1)(l2)，等号右侧各项是坐标或 node，各项之间不能加逗号），生成的盒子将等号右侧各项包围在内。各项与盒子边界的距离用选项 inner sep= 设置。

1.3 平面几何绘图举例

1.3.1 Elements' Book I, Proposition I

本小节展示的代码绘制下面的图形



Proposition I

To construct an equilateral triangle on a given finite straight line.

Let *AB* be the given finite straight line. ...

```
\begin{tikzpicture}[thick,help lines/.style={thin,draw=black!50}]
\def\A{\textcolor{input}{$A$}}
\def\B{\textcolor{input}{$B$}}
\def\C{\textcolor{output}{$C$}}
\def\D{$D$}
\def\E{$E$}
\colorlet{input}{blue!80!black}
\colorlet{output}{red!70!black}
\colorlet{triangle}{orange}
```

1 行，选项 thick 定义环境中的路径的线宽，help lines/.style= 定义 help lines 的样式。

2 行，\def 开启定义功能，\A 是定义的名称，{\textcolor{input}{A}} 是\A 的内容。

7 行，\colorlet 开启定义颜色功能，{input} 是颜色名称，{blue!80!black} 是定义的颜色。

```
\coordinate [label=left:\A] (A)
           at ($ (0,0) + .1*(rand,rand) $);
\coordinate [label=right:\B] (B)
           at ($ (1.25,0.25) + .1*(rand,rand) $);
\draw [input] (A) -- (B);
\node [name path=D,help lines,draw,label=left:\D] (D)
      at (A) [circle through=(B)] {};
\node [name path=E,help lines,draw,label=right:\E] (E)
      at (B) [circle through=(A)] {};
\path [name intersections={of=D and E,by={[label=above:\C]C}}];
\draw [output] (A) -- (C) -- (B);
\foreach \point in {A,B,C}
\fill [black,opacity=.5] (\point) circle (2pt);
```

10 行，\coordinate 开启定义坐标功能，选项 label=left:\A 指示给坐标加标签，标签在坐标的 left 侧,标签内容是 2 行定义的“\A”;括号 (A) 是定义的坐标的名称,at 指示坐标的位置;($ (0,0) + .1*(rand,ra 用向量运算定义一个点，其中 rand 是-1 到 1 之间的随机数，注意分数或小数与 rand 相乘必须加 * 符号，两个美元符号表示计算，而不是 LaTeX 的数学模式。

14 行，画线段 AB。

15 行，\node 开启 node（加标签）功能；选项 name path=D 定义 node 的边界是名称为 D 的路径；选项 help lines 调用 1 行定义的 help lines 样式；选项 draw 指示画出边界路径；括号 (D) 是该 node 的名称。

16 行,at 指示 15 行的 node 所指向的对象位置;(A) 是 node 所指向的对象;(A) [circle through=(B)] 画以 (A) 为圆心，经过点 (B) 的圆，需要调用 through 程序库；花括号 {} 内是 node 的内容，尽管无内容，但不能没有花括号。

19 行，\path 开启定义路径功能；name intersections= 用来定义两个路径的交点；of=D and E 指示分析路径 D 和 E 的交点；by= 指示用 C 来做交点名称，而 C 带有标签{[label=above:\C]C}，注意必须用花括号把[label=above:\C]C 括起来。

21 行，\foreach 开启 foreach 操作模式，对数组{A,B,C} 内的每个元素（用\point 代表）进行相同的操作。

22 行，即 21 行\foreach 指向的操作。\fill 指示填充颜色；选项 opacity=.5 设置不透明度为 0.5；circle 命令指示以 \point 为圆心，以 2pt 为半径画圆；\fill 以半透明的黑色填充该圆，实际上是描三点 A，B，C。

```
23 \begin{pgfonlayer}{background}
24 \fill[triangle!80] (A) -- (C) -- (B) -- cycle;
25 \end{pgfonlayer}
26 \node [below right, text width=5cm,align=justify] at (4,3)
27     {\small\textbf{Proposition I} \par
28         \emph{To construct an
29             \textcolor{triangle}{equilateral triangle}
30             on a given
31             \textcolor{input}{finite straight line}.}
32         \par\vskip1em
33       Let \A\B\ be the given
34       \textcolor{input}{finite straight line}. \dots
35      };
36 \end{tikzpicture}
```

23 行，在 background 层上作图。

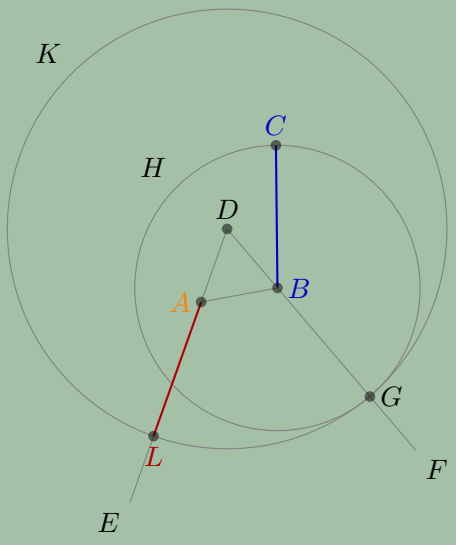
25 行，结束 background 层上作图。

26 行，给 (4,3) 点加标签，内容是一段文字。below right 指示标签在点 (4,3) 的右下方；text width=5cm 设置每行文字宽度为 10cm；align=justify 设置文字对齐方式是自适应方式。

1.3.2 Elements' Book I, Proposition II

本小节展示的代码绘制下面的图形

```
1 \begin{tikzpicture}[thick,help lines/.style={thin,draw=black!50}]
2 \def\A{\textcolor{orange}{$A$}}
3 \def\B{\textcolor{input}{$B$}}
4 \def\C{\textcolor{input}{$C$}}
5 \def\D{$D$}
6 \def\E{$E$}
7 \def\F{$F$}
```




```
\def\G{$G$}
\def\H{$H$}
\def\K{$K$}
\def\L{\textcolor{output}{$L$}}
\colorlet{input}{blue!80!black}
\colorlet{output}{red!70!black}
\coordinate [label=left:\A] (A)
            at ($ (0,0) + .1*(rand,rand) $);
\coordinate [label=right:\B] (B)
            at ($ (1,0.2) + .1*(rand,rand) $);
\coordinate [label=above:\C] (C)
            at ($ (1,2) + .1*(rand,rand) $);
```

以上主要是为后面画图作定义。

```
\draw [input] (B) -- (C);
\draw [help lines] (A) -- (B);
\coordinate [label=above:\D] (D)
     at ($ (A)!.5!(B) ! {sin(60)*2} ! 90:(B) $);
\draw [help lines] (D) -- ($ (D)!3.75!(A) $)
       coordinate [label=-135:\E] (E);
\draw [help lines] (D) -- ($ (D)!3.75!(B) $)
       coordinate [label=-45:\F] (F);
\node (H) at (B)
     [name path=H,help lines,circle through=(C),
       draw,label=135:\H] {};
\path [name path=B--F] (B) -- (F);
\path [name intersections={of=H and B--F,
                        by={[label=right:\G]G}}];
```

```
\node (K) at (D)
     [name path=K,help lines,circle through=(G),
       draw,label=135:\K] {};
\path [name path=A--E] (A) -- (E);
\path [name intersections={of=K and A--E,
                        by={[label=below:\L]L}}];
\draw [output] (A) -- (L);
\foreach \point in {A,B,C,D,G,L}
\fill [black,opacity=.5] (\point) circle (2pt);
\end{tikzpicture}
```

23 行，表达式($ (A)!x!(B) $) 确定直线 AB 上一个点，如果 x 在 0 到 1 之间，该点在线段 AB 内部；如果 x 大于 1，该点在 AB 外部；如果 x 是正数，则该点与 A、B 的距离之比为 x；如果 x 是负数，则该点是点($ (A)!|x|!(B) $) 关于点 A 的对称点。

表达式($ (X) ! {sin(60)*2} ! 90:(B) $) 指示以 X 为基点，将线段 XB 绕 X 旋转 90 度，再伸长 sin(60)*2 倍后，得到的线段端点。

表达式($ (A) ! .5 ! (B) ! {sin(60)*2} ! 90:(B) $) 指示先运算 ($ (A)!.5!(B) $) 得到一个点，然后该点参与后面的运算。

2 Tikz 的绘图环境和命令

在 LaTeX 中，要用 tikz 作图，首先调用 tikz 宏包和有关的程序库

```
\usepackage{tikz}
\usetikzlibrary{<list of libraries>}
```

然后开启 {tikzpicture} 环境作图，或者用命令 \tikz 开始作图。

各种命令必须以分号结束，否则报错。

2.1 {tikzpicture} 环境或 tikz 命令的格式

```
环境格式
\begin{tikzpicture}[<options>]
<environment contents>
\end{tikzpicture}

命令格式
\tikz[<options>]{<path commands>}
例如，\tikz{\draw (0,0) rectangle (2ex,1ex);}
```

2.2 主要的绘图对象

绘图时涉及的主要对象有

- 点

  符号 (1,2) 指定笛卡尔坐标点，默认坐标单位长度是 1cm。

  符号(30:1cm) 指定极坐标点，其中 30 表示 30°。

- 线

  主要有直线和曲线。

- Path

  以命令 \path 开头的句子创建一个 path 对象，构成路径的元素可以是点，线，node，图形等，不仅仅只有点和线。

- Coordinate

  以命令 \coordinate 开头的句子将某个对象（可以是点，线，node，图形等）设置为 coordinate 对象。注意点（数值坐标）与 coordinate 对象不同，coordinate 对象可以带多个选项使得其本身的内容更加丰富，而点则不能。例如 1.3.1 中例子的第 10 行代码所示。还有，变换选项 shift，rotate 对 coordinate 名称不起作用，而对点起作用。

- Node

  以命令 \node 开头的句子将某个对象（可以是点，线，node，图形等）设置为 node 对象，可以为 node 对象设置位置，边界形状，文字，颜色，标签等属性。

注意，创建了以上对象不等于显示它们，要显示创建的对象需要给这些命令添加选项 draw，或者用 \draw 开头的语句来绘图。

\draw <path>; 相当于 \path [draw] ...;。

注意，下面的语句不能正常编译

```
\path [name path=F] (0,0) -- (1,1);
\draw F;
```

也就是说，命令 \draw 后不能是路径名称。下面的语句是合适的

```
\path [name path=F,draw] (0,0) -- (1,1);
```

2.3 命令 \def

命令 \def 用于自定义一个对象，该对象名称以反斜线 \开头。句法是

```
\def<对象名称>{定义内容};
```

下面的例子定义 \wall （需要调用 pattern 程序库）

```
\begin{tikzpicture}
\def\wall{ \fill [fill=black!50] (1,-.5) rectangle (2,.5);
        \pattern [pattern=bricks] (1,-.5) rectangle (2,.5);
        \draw [line width=1pt] (1cm+.5pt,-.5) -- ++(0,1); }
\wall
\end{tikzpicture}
```

2.4 创建 coordinate 对象的句法及其命名

创建 coordinate 对象的句法是

```
\path . . . coordinate[<options>](<name>)at(<coordinate>) . . . ;
将路径中的某个对象设置为 coordinate 对象，这个对象可以是点，线，node，标签等等。这与下句等价
\node[shape=coordinate][<options>](<name>)at(<coordinate>){};
```

```
\coordinate [<options>](<name>)at(<coordinate>);
这是上一句法的简写，不要在该命令后面用 node 语句给坐标加标签，可在方括号里用 label 选项。
```

例如，在 1.3.1 的例子中有下面语句

```
\draw[->] (-1.5,0) -- (1.5,0) node[right] {$x$}
      coordinate(x axis);
\coordinate [label=left:\A] (A)
         at ($ (0,0) + .1*(rand,rand) $);
\draw [help lines] (D) -- ($ (D)!3.75!(A) $)
      coordinate [label=-135:\E] (E);
```

2.5 Bounding Box

盒子是一个占据一定空间的对象，盒子有宽度（width），基线（baseline），基线到盒子的上界是高度（height），基线到盒子下界是深度（depth）。默认基线是图形的 x 轴。

将绘制的图形作为盒子，可用如下的选项或命令

```
选项 use as bounding box
命令 \useasboundingbox，是 \path[use as bounding box] 的简写。
```

```
Left of picture\begin{tikzpicture}
\draw[use as bounding box]
     (2,0) rectangle (3,1);
\draw (1,0) -- (4,.75);
\end{tikzpicture}right of picture.
```

Left of picture right of picture.

```
Left of picture
\begin{tikzpicture}
\useasboundingbox (0,0) rectangle (3,1);
\fill (.75,.25) circle (.5cm);
\end{tikzpicture}
right of picture.
```

Left of picture right of picture.

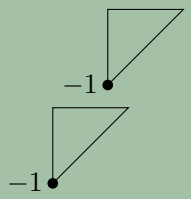
选项current bounding box
选项current path bounding box
这两个选项可以把当前的图形或者路径看作矩形盒子。

trim left=<dimension or default> 该选项将图形向左平移<dimension> 的距离，
即将盒子之外左侧的内容向右移来覆盖盒子，默认 0pt。
trim right=<dimension> 意义类似，只是没有默认值。

```
Text before image.
\begin{tikzpicture}[trim left, trim right=2cm, baseline]
\draw (-1,-1) grid (3,2);
\fill (0,0) circle (5pt);
\end{tikzpicture}
```

Text before image. Text after image.

```
\begin{tikzpicture}
\draw (0,1) -- (0,0) -- (1,1) -- cycle;
\fill (0,0) circle (2pt);
\node[left] at (0,0) {$-1$};
\end{tikzpicture}
\par
\begin{tikzpicture}[trim left]
\draw (0,1) -- (0,0) -- (1,1) -- cycle;
\fill (0,0) circle (2pt);
\node[left] at (0,0) {$-1$};
\end{tikzpicture}
```



2.6 设置图的底线 baseline 的位置

通常，图的下端底线与两侧文字的基线平齐。baseline 选项设置图的底线与文字基线之间的距离。

```
baseline=<dimension or coordinate or default> 该选项的 dimension 从 x 轴算起，
默认长度单位为 pt。若 dimension 为负数则基线从 x 轴下移。默认以 x 轴为基线。
coordinate 该选项指定基线通过该坐标。
baseline=default 使用默认值，基线在 x 轴上。
```

```
前面的选项 trim left，trim right 设置图形的左右边界，
这里 baseline 调整图形的上下位置。
```

```
Top align:
\tikz[baseline=(current bounding box.north)]
\draw (0,0) rectangle (1cm,2ex);
```

Top align: 

2.7 用 scope 环境或 scoped 命令嵌入图形

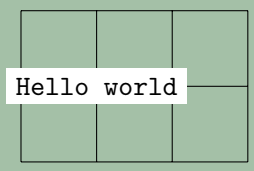
用这个环境或命令可以在 {tikzpicture} 环境或 \tikz 命令之内嵌入子图形。

```
环境格式
\begin{scope}[<options>]
<environment contents>
\end{scope}

命令格式
\usetikzlibrary{scopes} % 在导言区调用 scopes 程序库
\scoped[<options>]<path command>
```

例如

```
\begin{tikzpicture}
\node [fill=white] at (1,1) {Hello world};
\scoped [on background layer]
\draw (0,0) grid (3,2);
\end{tikzpicture}
```

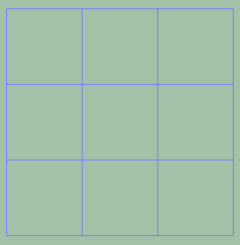


2.8 全局选项设置语句以及 /.style 与 /.default 的设置

```
\tikzset{<options>} 全局设置，影响整个分组
my style/.style={<options>} 定义名称为 my style 的样式
hkeyi/.default=<value> 设置名称为 hkeyi 的对象的默认值
```

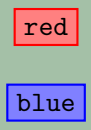
例如，网格对象 help lines 是预定义的，其网格线宽的初始值是 line width=0.2pt,gray，可以用上述选项修改它。

```
\tikzset{help lines/.style=very thin} % 设置线宽为 very thin。
\tikzset{Karl' s grid/.style={help lines,color=blue!50}}
% 定义名称为 Karl' s grid 的样式，样式内容是网格 help lines，
% 并设置网格线的颜色是 50% 的蓝色。
\draw[Karl' s grid] (0,0) grid (3,3);
```

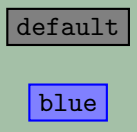


“/.style” 的选项可以带一个变量，用 #1 表示，例如

```
\begin{tikzpicture}[outline/.style={draw=#1,thick,fill=#1!50}]
\node [outline=red] at (0,1) {red};
\node [outline=blue] at (0,0) {blue};
\end{tikzpicture}
```




```
\begin{tikzpicture}[outline/.style={draw=#1,thick,fill=#1!50},
outline/.default=black]
\node [outline] at (0,1) {default};
\node [outline=blue] at (0,0) {blue};
\end{tikzpicture}
```



2.9 过指定点的水平线和竖直线

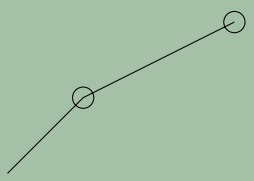
```
在导言区调用through程序库 \usetikzlibrary{through}，句法
horizontal line through={(<coordinate>)}
vertical line through={(<coordinate>)}
```

3 Path 的相关操作

3.1 设置路径样式，在路径中插入路径

命令 \path 将它后面的内容定义为路径，它只能用在{tikzpicture}环境中。
every path/.style={样式定义} 设置环境中所有路径共同的样式（每个路径可以作本地修改）。
insert path=<path> 给路径内部的某个位置添加路径。

```
\tikz [c/.style={insert path={circle[radius=4pt]}}]
\draw (0,0) -- (1,1) [c] -- (3,2) [c];
```



3.2 三种路径连接方式以及环路

Move-To，从路径的前一部分到后一部分没有连线。

Line-To，用直线段连接路径的前一部分与后一部分，用两个短线“--”表示。

Curve-To，用曲线（贝塞尔曲线）连接路径的前一部分与后一部分，用两个点“..”表示。

--cycle 是环路标识符号。

控制曲线采用如下格式

```
.. controls <first control point> and <second control point> .. <end point>
```

例如

```
\draw (0,0) .. controls (1,1) and (2,1) .. (2,0);
```

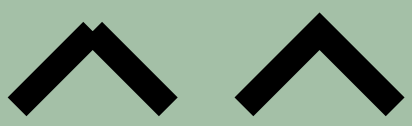
```
\draw (-1,0) .. controls (-1,0.555) and (-0.555,1) .. (0,1)
.. controls (0.555,1) and (1,0.555) .. (1,0);
```

曲线也可用以连接 node 的位置

```
\draw [->] (waiting.west) .. controls +(left:5mm) and +(up:5mm)
.. (enter critical.north);
```

三种路径连接方式在交接点处有不同的表现，下面是 Move-To 和 Line-To 的区别

```
\begin{tikzpicture}[line width=10pt]
\draw (0,0) --(1,1) (1,1) --(2,0);
\draw (3,0) -- (4,1) -- (5,0);
\end{tikzpicture}
```



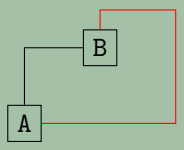
环路对交接点处的影响如下例

```
\begin{tikzpicture}[line width=10pt]
\draw (0,0) -- (1,1) -- (1,0) -- (0,0)
     (2,0) -- (3,1) -- (3,0) -- (2,0);
\draw (5,0) -- (6,1) -- (6,0) -- cycle
     (7,0) -- (8,1) -- (8,0) -- cycle;
\end{tikzpicture}
```

3.3 用横竖线连接路径部分

“<位置1> | - <位置2>”表示用过 <位置1> 的竖线和过 <位置2> 横线把这两个位置连接起来，与 <位置2> - | <位置1> 效果一样。这里的 <位置> 可以是坐标，也可以是锚位置等。

```
\begin{tikzpicture}
\draw (0,0) node(a) [draw] {A} (1,1) node(b) [draw] {B};
\draw (a.north) |- (b.west);
\draw[color=red] (a.east) -| (2,1.5) -| (b.north);
\end{tikzpicture}
```



3.4 连线算子 to 和 edge

参考 node。

3.5 常用路径选项，命名、颜色、线型等

```
name path=<name> 给路径命名
```

```
color 路径使用默认颜色
color=<color name> 给路径设置颜色
```

```
draw 画出路径
draw=<color name> 定义画笔颜色
draw=none 不画路径
```

```
line width=<dimension> 定义线宽，初始值 0.4pt
```

```
预定义的线宽，由细到粗依次是
ultra thin，very thin，thin，semithick，thick，very thick，ultra thick
```

```
line cap=<type> 定义线端头样式，初始值 butt，可选round，rect 或 butt
```

line join=<type> 定义两线交接处的样式，初始值 miter，可选round，bevel，miter
miter limit=<factor> 初始值 10

dash pattern=<dash pattern>，dash phase=<dash phase>，
预定义的线条样式
solid，dotted，densely dotted，loosely dotted，dashed，densely dashed，loosely dashed，
dash dot，densely dash dot，loosely dash dot，dash dot dot，densely dash dot dot，
loosely dash dot dot

double 设置路径为双线
double=<core color> 设置双线中的第二条线的颜色
double distance=<dimension> 设置双线的间距

fill=<color> 设置填充区域的颜色
fill=none 不填充颜色

pattern=<name> 设置样式线条，需要调用 patterns 程序库
pattern color=<color> 设置样式线条的颜色

decorate 启用装饰样式，需调用相应程序库
decoration=<style> 设置装饰样式，需调用 decoration 程序库

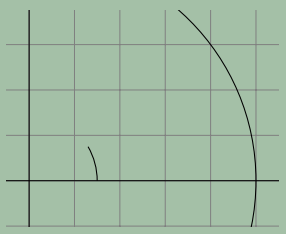
shade 启用颜色渐变效果
shading=<name> 设置颜色渐变的样式，需要调用 shadings 程序库。

shadow 启用阴影，需要调用 shadows 程序库

3.6 剪贴命令 clip

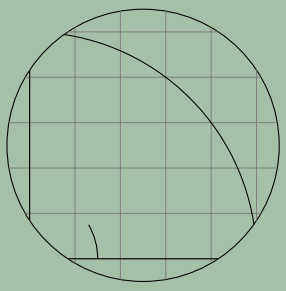
用矩形剪贴

```
\begin{tikzpicture}[scale=3]
\clip (-0.1,-0.2) rectangle (1.1,0.75);
\draw[step=.2cm,gray,very thin] (-1.4,-1.4) grid (1.4,1.4);
\draw (-1.5,0) -- (1.5,0);
\draw (0,-1.5) -- (0,1.5);
\draw (0,0) circle [radius=1cm];
\draw (3mm,0mm) arc [start angle=0, end angle=30, radius=3mm];
\end{tikzpicture}
```



用圆形剪贴

```
\begin{tikzpicture}[scale=3]
\clip[draw] (0.5,0.5) circle (.6cm);
\draw[step=.2cm,gray,very thin] (-1.4,-1.4) grid (1.4,1.4);
\draw (-1.5,0) -- (1.5,0);
\draw (0,-1.5) -- (0,1.5);
\draw (0,0) circle [radius=1cm];
\draw (3mm,0mm) arc [start angle=0, end angle=30, radius=3mm];
\end{tikzpicture}
```



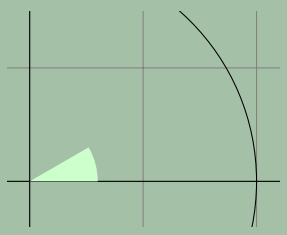
3.7 填充颜色

颜色填充规则

- nonzero rule，这是默认填色规则；为了确定点 A 是否属于被填充的区域，考虑从 A 出发的射线 $\vec{l}$，并从 0 开始计数；如果 $\vec{l}$ 与区域的边界相交且边界从 $\vec{l}$ 左侧穿到右侧，则计数加 1，否则计数减 1；若计数结果非 0，则点 A 属于该区域，否则不属于该区域。
- even odd rule，与上一规则做法类似，如果 $\vec{l}$ 与区域边界相交的次数是奇数，则点 A 属于该区域，否则不属于该区域。

用 \fill 命令填充颜色

```
\begin{tikzpicture}[scale=3]
\clip (-0.1,-0.2) rectangle (1.1,0.75);
\draw[step=.5cm,gray,very thin]
     (-1.4,-1.4) grid (1.4,1.4);
\draw (-1.5,0) -- (1.5,0);
\draw (0,-1.5) -- (0,1.5);
\draw (0,0) circle [radius=1cm];
\fill[green!20!white] (0,0) -- (3mm,0mm)
    arc [start angle=0, end angle=30, radius=3mm]
     -- (0,0);
\end{tikzpicture}
```

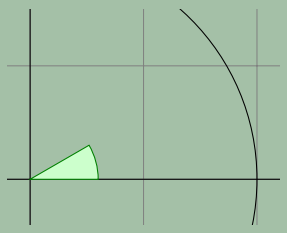


符号 green!20!white 的意思是绿色占 20%、白色占 80%。上面最后的代码可以用 -- cycle 结尾:

```
\fill[green!20!white] (0,0) -- (3mm,0mm)
     arc [start angle=0, end angle=30, radius=3mm] -- cycle;
```

\filldraw 命令先画出路径，然后填充颜色

```
\begin{tikzpicture}[scale=3]
\clip (-0.1,-0.2) rectangle (1.1,0.75);
\draw[step=.5cm,gray,very thin] (-1.4,-1.4) grid (1.4,1.4);
\draw (-1.5,0) -- (1.5,0);
\draw (0,-1.5) -- (0,1.5);
\draw (0,0) circle [radius=1cm];
\filldraw [fill=green!20!white, draw=green!50!black]
        (0,0) -- (3mm,0mm)
        arc [start angle=0, end angle=30,
           radius=3mm] -- cycle;
\end{tikzpicture}
```



3.8 颜色渐变

颜色渐变命令\shade 和\shadedraw

```
\tikz \shade (0,0) rectangle (2,1) (3,0.5) circle (.5cm);
```

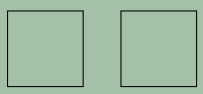
默认的是由灰色到白的渐变。如下选择颜色渐变

```
\begin{tikzpicture}[rounded corners,ultra thick]
\shade [top color=yellow,bottom color=black]
      (0,0) rectangle +(2,1);
\shade [left color=yellow,right color=black]
      (3,0) rectangle +(2,1);
\shadedraw [inner color=yellow,outer color=black,
         draw=yellow] (6,0) rectangle +(2,1);
\shade[ball color=green] (9,.5) circle (.5cm);
\end{tikzpicture}
```

3.9 用 + 表示的平移

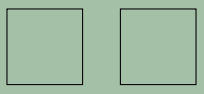
平移 ++(acm, bcm) 是将紧邻此符号之前的点加上向量 (acm, bcm)

```
\begin{tikzpicture}
\def\rectanglepath{-- ++(1cm,0cm) -- ++(0cm,1cm) --
                ++(-1cm,0cm) -- cycle}
\draw (0,0) \rectanglepath;
\draw (1.5,0) \rectanglepath;
\end{tikzpicture}
```



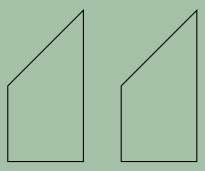
上面的小方形可以简化为:

```
\tikz \draw (0,0) rectangle +(1,1) (1.5,0) rectangle +(1,1);
```



平移 +(acm, bcm) 是将最初的（不是紧邻的）起始点加上向量 (acm, bcm)

```
\begin{tikzpicture}
\def\rectanglepath{-- +(1cm,0cm) -- +(1cm,2cm) --
                   +(0cm,1cm) -- cycle}
\draw (0,0) \rectanglepath;
\draw (1.5,0) \rectanglepath;
\end{tikzpicture}
```



4 由两点确定第三点，基本的坐标计算

在导言区设置 \usetikzlibrary{calc} 调用程序库 calc，然后对坐标做基本计算。

4.1 坐标计算格式

句法格式

```
([<options>]$<coordinate computation>$)
```

这里要用圆括号和美元符号把坐标运算式子包起来。数乘向量必须加 * 号。如果不调用 math 程序库，坐标运算式中只允许用小数作加、减、乘运算，更复杂的运算，比如，分数，开方，用函数表达式表示的值都是无法计算的。调用 math 程序库后，可以使用程序库允许的运算结构。

```
<coordinate>!<number>!<angle>:<second coordinate>
```

其中“!<angle>:”可以没有；但如果有“!<angle>:”，则“!<number>”必须有。注意感叹号“!”、冒号“:”两侧不能有空格。

4.2 用比例、旋转确定第三点

表达式 ($ (A)!x!(B) $) 确定直线 AB 上一个点，该点与点 A 的距离比上 AB 的长度就是 x；如果 x 大于 0，该点在射线 AB 上；如果 x 小于 0，该点在射线 BA 上，且点 A 介于该点与点 B 之间。

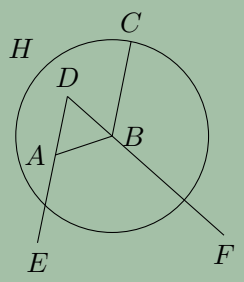
表达式 ($ (X)!{sin(60)*2}!90:(B) $) 指示以 X 为基点，将线段 XB 绕 X 旋转 90 度，再伸长 sin(60)*2 倍后，得到的线段端点。

表达式 ($ (A)! .5!(B)!{sin(60)*2}!90:(B) $) 指示先运算 ($ (A)!.5!(B) $) 得到一个点，然后该点参与后面的运算。

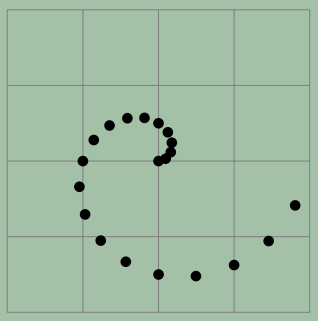
4.3 垂足

表达式 ($ (A)!(B)!(C) $) 指示点 B 在直线 AC 上的正交射影点（垂足），垂足可以在线段 AC 之外。

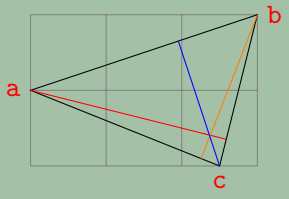
```
\begin{tikzpicture}
\coordinate [label=left:$A$] (A) at (0,0);
\coordinate [label=right:$B$] (B) at (0.75,0.25);
\coordinate [label=above:$C$] (C) at (1,1.5);
\draw (A) -- (B) -- (C);
\coordinate [label=above:$D$] (D)
          at ($ (A) ! .5 ! (B) ! {sin(60)*2} ! 90:(B) $) {};
\node (H) [label=135:$H$,draw,circle through=(C)] at (B) {};
\draw (D) -- ($ (D) ! 3.5 ! (B) $)
           coordinate [label=below:$F$] (F);
\draw (D) -- ($ (D) ! 2.5 ! (A) $)
           coordinate [label=below:$E$] (E);
\end{tikzpicture}
```




```
\begin{tikzpicture}
\draw [help lines] (0,0) grid (4,4);
\foreach \i in {0,0.1,...,2}
\fill ($(2,2) !\i! \i*180:(3,2)$) circle (2pt);
\end{tikzpicture}
```



```
\begin{tikzpicture}
\draw [help lines] (0,0) grid (3,2);
\coordinate (a) at (0,1);
\coordinate (b) at (3,2);
\coordinate (c) at (2.5,0);
\draw (a) node[red,left]{a} -- (b) node[red,right]{b} --
     (c) node[red,below]{c} -- cycle;
\draw[red] (a) -- ($(b)!(a)!(c)$);
\draw[orange] (b) -- ($(a)!(b)!(c)$);
\draw[blue] (c) -- ($(a)!(c)!(b)$);
\end{tikzpicture}
```



4.4 用横竖 - | 算子

符号 (30:1cm |- 0,0) 表示过点 (30:1cm) 的竖直线与过 (0cm,0cm) 点的水平线的交点。(2,1 |- 3,4) 与 (3,4 -| 2,1) 所定义的点相同。注意其中坐标没有括号。

5 交点

调用 intersections 程序库，可对交点设置多个选项。选项 name path=<name> 给当前画的路径命名，用选项 name intersections=<options> 给两个路径的交点命名。

name intersections 也可带选项

```
name intersections={of=<path1> and <path2>, name=<prefix>,
            total=<macro>, by={<comma-separated list>},
             sort by=<path name>}
```

of= 指定两个路径。

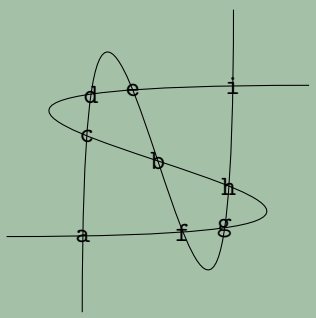
name= 给交点命名（交点的共名），如果 name=i，则这些交点就可用且只能用 i-1，i-2，i-3 这种带编号的形式来引用，引用编号从 1 开始（不能只用无编号的 i 引用，即使只有一个交点）。

total=macro，macro 是以命令符号 \开头的一组符号，代表交点的总数目。

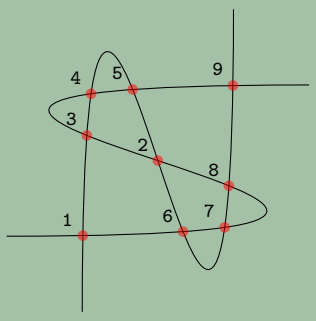
sort by=，按指定路径的方向将交点排序。

by={comma-separated list}，一组符号或构造对象（个数随意），之间用逗号隔开，按次序将列举的符号或对象赋予交点（可以用列出的符号引用交点，故也相当于命名，有时要比用 name= 选项命名更方便）；用方括号把 comma-separated list 中列举的对象括起来，方括号内用省略号...，可以按规律自动生成想要列举项目。

```
\begin{tikzpicture}
\clip (-2,-2) rectangle (2,2);
\draw [name path=curve 1] (-2,-1) .. controls (8,-1)
                          and (-8,1) .. (2,1);
\draw [name path=curve 2] (-1,-2) .. controls (-1,8)
                          and (1,-8) .. (1,2);
\fill [name intersections={
          of=curve 1 and curve 2,
          by={[label=center:a],[label=center:...],
              [label=center:i]}}];
\end{tikzpicture}
```

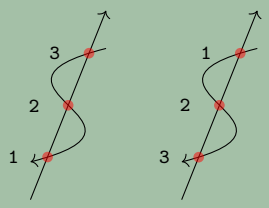


```
\begin{tikzpicture}
\clip (-2,-2) rectangle (2,2);
\draw [name path=curve 1] (-2,-1) .. controls (8,-1)
                          and (-8,1) .. (2,1);
\draw [name path=curve 2] (-1,-2) .. controls (-1,8)
                          and (1,-8) .. (1,2);
\fill [name intersections={of=curve 1 and curve 2,
       name=i,      total=\t}]
     [every node/.style={above left, black, opacity=1},
       red, opacity=0.5]
\foreach \s in {1,...,\t}{(i-\s) circle (2pt)
          node {\footnotesize\s}};
\end{tikzpicture}
```

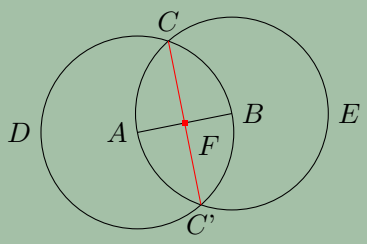


下图中用了选项 sort by=\pathname，所以画了两个图

```
\begin{tikzpicture}
\clip (-0.5,-0.75) rectangle (3.25,2.25);
\foreach \pathname/\shift in {line/0cm, curve/2cm}{
\tikzset{xshift=\shift}
\draw [->, name path=curve] (1,1.5) .. controls (-1,1)
                           and (2,0.5) .. (0,0);
\draw [->, name path=line] (0,-.5) -- (1,2) ;
\fill [name intersections={of=line and curve,
       sort by=\pathname, name=i}]
     [every node/.style={left=.25cm, black, opacity=1},
       red, opacity=0.5]
\foreach \s in {1,2,3}{(i-\s) circle (2pt)
       node {\footnotesize\s}};
}
\end{tikzpicture}
```



```
\begin{tikzpicture}
\coordinate [label=left:$A$] (A) at (0,0);
\coordinate [label=right:$B$] (B) at (1.25,0.25);
\draw [name path=A--B] (A) -- (B);
\node (D) [name path=D,draw,circle through=(B),
           label=left:$D$]
         at (A) {};
\node (E) [name path=E,draw,circle through=(A),
         label=right:$E$]
         at (B) {};
\path [name intersections={of=D and E,
      by={[label=above:$C$]C, [label=below:$C' $]C' } }];
\draw [name path=C--C' ,red] (C) -- (C' );
\path [name intersections={of=A--B and C--C' ,by=F}];
\node [fill=red,inner sep=1pt,label=-45:$F$] at (F) {};
\end{tikzpicture}
```



6 重复操作

用含 foreach 的语句实现重复操作，句法

```
\path . . . foreach <variables> [options] in {path commands} . . . ;
上面的 foreach 可用\foreach 代替
```

```
\foreach \x in {1,2,3} {$x =\x$, }
```

$x = 1,\ x = 2,\ x = 3,$

```
\tikz \foreach \x in {1,...,10}
\draw (\x,0) circle (0.4cm);
```

下面的数组是以前两项之差为公差的等差数列

```
\tikz \foreach \x in {-1,-0.5,...,8}
\draw (\x cm,-1pt) -- (\x cm,1pt);
```

```
\tikz \draw (0,0) foreach \x in {1,...,3} { -- (\x,1) -- (\x,0) };
```

7 node 绘图

7.1 node 的句法

```
\path . . . (<coordinate>) node [<options>](<name>)
          {<node contents>} . . . ;
```

node 前面的坐标是它指向的对象，方括号里是选项，圆括号 (<name>) 里是 node 的名称（可以不命名），花括号里是内容，但即使无内容也不能省略花括号，选项中含 node contents=<node contents> 时可省略花括号及其中的内容。

在调用中文宏包后，可以直接用汉字给 node 命名，也可以直接用汉字填写 node 的内容。可以用 (4)、(4-1)、(4-1-2)、(4-1+0)、(4-1.0)、(韶)、(韶-0) 这种形式给 node 命名。但不能用 (4.1)、(4.1-0) 这种第一个标点是点号、逗号、冒号的名称。合适的名称都可以用在 \draw 命令之后。

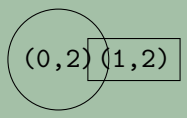
可用 at 指示 node 指向的位置。

```
\path . . . node [<options>](<name>)
              at(<coordinate>){<node contents>} . . . ;
```

可以将 \path . . . node 简略为 \node 命令，其中 at 可省略

```
\node [<options>](<name>) (<coordinate>)
          {<node contents>} . . . ;
```

```
\begin{tikzpicture}
\path (0,2) node [shape=circle,draw] {(0,2)}
      ( 1,2) node [draw] {(1,2)};
\end{tikzpicture}
```




```
\begin{tikzpicture}
\node at ( 0,2) [circle,draw] {(0,2)};
\end{tikzpicture}
```

7.2 node 的常用选项

shape=<shape name> 选择 coordinate，circle，ellipse，diamond，dart 等形状。
初始形状是 rectangle，“shape=” 可省略;
如果名称为 x 的 node 选择了 coordinate 形状，就可以把它当作 coordinate (x)
来利用，即把它看成坐标。

draw 画出形状的边界

fill 填充颜色

label 给 node 加标签

behind path 指示路径可能会遮挡 node （如果二者有重叠）
in front of path 意思与上相反

double 边界线是双线

rounded corners 边界上的角是圆角

rounded corners=<dimension> 设置边界圆角的半径

inner sep=<dimension> 设置 node 的文字内容与边界的距离

minimum height=<dimension> 设置 node 的最小高度

minimum width=<dimension>
minimum size=<dimension>
shape aspect=<aspect ratio> 设置 node 的宽高之比

shape border rotate=<angle> 使得 node 按指定角度旋转

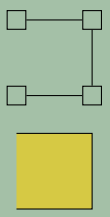
name=<name> 设置 node 的名称，可以改为上面句法中的命名方法

下面三句等效

```
\node[shape=coordinate][<options>](<name>)at(<coordinate>){};
\path . . . coordinate[<options>](<name>)at(<coordinate>) . . . ;
\coordinate . . . ;
```

选项 shape=<shape> 的形状 coordinate 与有形的 rectangle 形状用很大区别。对 rectangle 形状使用裁线规则，即当该 node 与其他位置连线时，线的起点在矩形边界上。而对 coordinate 形状不使用裁线规则，即当该 node 与其他位置连线时，线的起点在坐标点上。而且对由 coordinate 形状构成的不封闭图形也可以填充颜色。

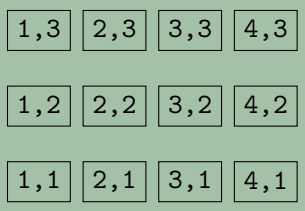
```
\begin{tikzpicture}[every node/.style={draw}]
\path[yshift=1.5cm,shape=rectangle]
      (0,0) node(a1){} (1,0) node(a2){}
      (1,1) node(a3){} (0,1) node(a4){};
\filldraw[fill=yellow!80!black]
      (a1) -- (a2) -- (a3) -- (a4);
\path[shape=coordinate]
      (0,0) coordinate(b1) (1,0) coordinate(b2)
      (1,1) coordinate(b3) (0,1) coordinate(b4);
\filldraw[fill=yellow!80!black]
      (b1) -- (b2) -- (b3) -- (b4);
\end{tikzpicture}
```



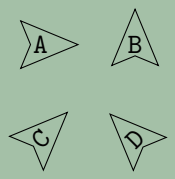
7.3 在 node 中使用 foreach 语句

举例如下

```
\tikz \node foreach \x in {1,...,4}
          foreach \y in {1,2,3}
          [draw] at (\x,\y) {\x,\y};
```




```
\tikzstyle{every node}=[dart, inner sep=1pt, draw]
\tikz \node foreach \a/\b/\c in
                {A/0/0, B/45/0, C/0/45, D/45/45}
          [shape border rotate=\b, rotate=\c]
          at (\b/36,-\c/36) {\a};
```

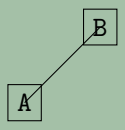


7.4 为环境内所有 node 设置样式

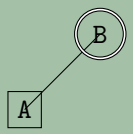
```
every <shape> node/.style={style}
```

环境的这个选项可以为环境内的每个形状为 <shape>（如果无此形状选项，就针对所有 node）的 node 设置相同的基本样式，各个 node 可以在这个基本样式的基础上再分别修改。

```
\begin{tikzpicture}[every node/.style={draw}]
\draw (0,0) node {A} -- (1,1) node {B};
\end{tikzpicture}
```



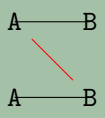

```
\begin{tikzpicture}
     [every rectangle node/.style={draw},
      every circle node/.style={draw,double}]
\draw (0,0) node[rectangle] {A} -- (1,1) node[circle] {B};
\end{tikzpicture}
```




```
name prefix=<text>
```

环境的这个选项可以为环境内的所有 node 的名称设置相同的前缀，即“共名”，以便于引用，观察下例，注意其中前缀之后加短线 -

```
\tikz {
      \begin{scope}[name prefix = top-]
      \node (A) at (0,1) {A};
      \node (B) at (1,1) {B};
      \draw (A) -- (B);
      \end{scope}
      \begin{scope}[name prefix = bottom-]
      \node (A) at (0,0) {A};
      \node (B) at (1,0) {B};
      \draw (A) -- (B);
      \end{scope}
      \draw [red] (top-A) -- (bottom-B);
       }
```



7.5 node 中的文字编辑

在 node 中的文字有多个选项

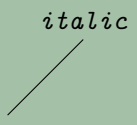
```
text=<color> 设置文字颜色
```

```
node font=<font commands> 设置 node 的所有字体
```

```
font=<font commands> 设置 node 的花括号内的字体
```

例如

```
\begin{tikzpicture}
\draw[node font=\itshape] (1,0) -- +(1,1) node[above] {italic};
\end{tikzpicture}
```



7.6 可以在 node 的内容中插入环境

例如插入 LaTeX 的 tabular 环境

```
\tikz \node [draw] {
\begin{tabular}{cc}
     upper left & upper right\\
     lower left & lower right
\end{tabular}
                 };
```

upper left	upper right
lower left	lower right

7.7 node 的文字对齐方式

可以用如下选项选择 node 内容中文字对齐方式

```
align=center
align=right
align=flush left
align=flush center
align=justify
```

```
\tikz[align=center] \node[draw] {This is a\\demonstration.};
```

This is a
demonstration.

```
\tikz \node[fill=yellow!80!black,align=right]
{This is a\\[-2pt] demonstration text for\\[1ex] alignments.};
```

This is a
demonstration text for
alignments.

```
\tikz \node[fill=yellow!80!black,text width=3cm,align=flush center]
{This is a demonstration text for showing how line breaking works.};
```

This is a
demonstration
text for
showing how
line breaking
works.

7.8 设置 node 每行文字的最大宽度

可用选项 text width=<dimension> 设置 node 每行文字的最大宽度

```
\tikz \draw (0,0) node[fill=yellow!80!black,text width=3cm]
{This is a demonstration text for showing how line breaking works.};
```

This is a
demonstration
text for
showing how line
breaking works.

7.9 node 的锚位置

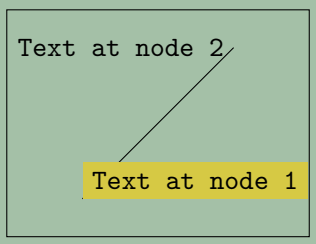
node 指向的对象可以是坐标、node、路径等，这些对象都有以下位置

- 中心，center
- 罗盘位置，north，north east，east，south east，south，south west，west，north west，罗盘位置都在对象的边界上
- 基线位置，base，base west，base east

- 中间水平线位置，mid，mid west，mid east
- 后缀标记方位,如果对象名称为 (u),还可用后缀标记方位 u.north,u.north east,u.south west，u.center 等

锚（anchor）位置选项的格式是 anchor=<anchor name>,指示 node 的“锚”放在对象的 <anchor name> 位置，注意 “锚与船的位置总是相对的”，即 node 的显示位置总与指定的锚位相对。

```
\begin{tikzpicture}
\draw (-0.5,0) rectangle (3.5,3);
\draw (0.5,0.5) node [fill=yellow!80!black, anchor=south west]
            {Text at \verb!node 1!}
      -- (2.5,2.5) node [anchor=east]
            {Text at \verb!node 2!};
\end{tikzpicture}
```



7.10 node 的平移位置

```
above 将 node 平移到所指定的对象上部，但仍然在对象的边界线上
below，left，right，
```

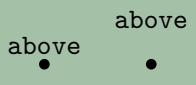
```
above=<offset> 设置从所指定的对象的中心向上平移的距离
below=<offset>，left=<offset>，right=<offset>
```

```
above left，above right，below left，below right
```

```
above left=距离1and距离2，从所指定的对象中心算起，上移距离1并左移距离2
above right=距离1and距离2
below left=距离1and距离2
below right=距离1and距离2
```

```
centered
```

```
\tikz \fill (0,0) circle (2pt) node[above] {above};
\tikz \fill (0,0) circle (2pt) node[above=10pt] {above};
```



7.11 设置诸 node 的相互位置

首先调用 positioning 程序库。

举例来说

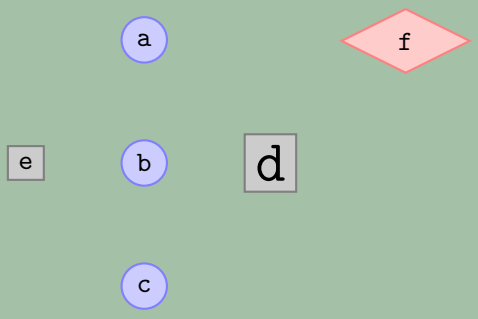
```
\node at (1,1) [above=2,draw] {above};
定义 node 的中心在 (1,1) 之上 2cm 处
```

```
\node at (1,1) [above right=.2 and 3mm,draw] {above};
定义 node 的中心在 (1,1) 之上0.2cm，之右3mm处
```

```
\node [above=5mm of somenode.north east]
定义 node 的中心在名为 somenode 的 node 的 north east 位置之上 5mm 处
```

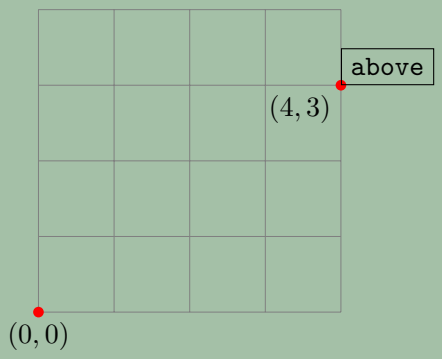
注意，above=of 表示按（默认的或设定的）距离 node distance （默认 1cm）上移，above=5mm of 表示上移 5mm，这里必须带 of，of 后的 node 名称不带括号。

```
\usetikzlibrary{positioning} % 在导言区设置
\begin{tikzpicture}
     [place/.style={circle,draw=blue!50,fill=blue!20,thick,
       inner sep=0pt,minimum size=6mm},
      transition/.style={rectangle,draw=black!50,fill=black!20,
       thick,inner sep=4pt,minimum size=4mm},
      display/.style={draw=red!50,fill=red!20,thick,
       inner sep=5pt},diamond,aspect=2]
\node[place] (a) at ( 0,2) {a};
\node[place] (b) [below=of a] {b};
\node[place] (c) [below=of b] {c};
\node[transition] (d) [right=of b] {\huge d};
\node[transition] (e) [left=of b] {e};
\node[display] (f) [above right=of d.north east] {f};
\end{tikzpicture}
```



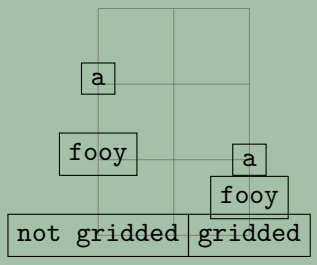
位置选项 above right=3 and 4 指示向上平移 3cm，向右平移 4cm; above=3 and 4 指示向上平移 3cm，没有横向平移，故 4 无用。

```
\begin{tikzpicture}
\draw[help lines] (0,0) grid (4,4);
\fill [red](0,0) circle [radius=2pt];
\fill [red](4,3) circle [radius=2pt];
\node (0,0) [below] {$(0,0)$};
\node (4,3) [below left]{$(4,3)$} ;
\node at (0,0) [above right=3 and 4,draw] {above};
\end{tikzpicture}
```



选项“node distance= 距离”确定两个 node 的间距。当环境带有 on grid 选项时，该值指的是 node 的中心之间的距离;当环境无 on grid 选项时,该值指的是 node 边界之间的距离,初始值为 1cm。

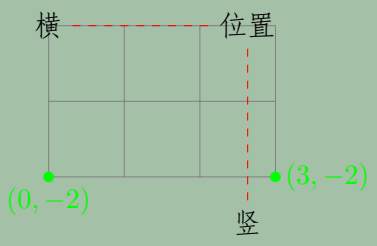
```
\begin{tikzpicture}[every node/.style=draw,node distance=5mm]
\draw[help lines] (0,0) grid (2,3);
% Not gridded
\node (a1) at (0,0) {not gridded};
\node (b1) [above=of a1] {fooy};
\node (c1) [above=of b1] {a};
% gridded
\begin{scope}[on grid]
\node (a2) at (2,0) {gridded};
\node (b2) [above=of a2] {fooy};
\node (c2) [above=of b2] {a};
\end{scope}
\end{tikzpicture}
```



7.12 使用横竖 - | 算子

类似于 (2,1 |- 3,4) 这种句法,也可用 (a - | b) 这种句法来确定一个位置,其中 a,b 是 node 名，且不加括号。

```
\begin{tikzpicture}
\draw[help lines] (0,0) grid (3,-2);
\node (横) at (0,0) {横};
\node (竖) [below right=2 and 2 of 横] {竖};
\node (位置) at (横-|竖){位置};
\draw [dashed,red](横) -- (位置) -- (竖);
\fill [green](0,-2) circle (2pt) node [below]{$(0,-2)$}
     (3,-2) circle (2pt) node [right]{$(3,-2)$};
\end{tikzpicture}
```



7.13 路径上 node 标签的位置选项

把 node 作为标签添加到路径上，通常，node 的位置取决于代码的次序

(0,0)node{a} -- (1,1) 把 node 标签加在(0,0)点处

(0,0) --node{a}(1,1) 把 node 标签加在线中间

(0,0) -- (1,1)node{a} 把 node 标签加在(1,1)点处

也可以用选项调整 node 标签的位置，常用的标签位置选项如下

pos=<小数>

auto，auto=<direction>
swap 或撇号 `

sloped

allow upside down=<boolean> 初始值 false

midway 等于 pos=0.5

near start 等于 pos=0.25

near end 等于 pos=0.75

very near start 等于 pos=0.125

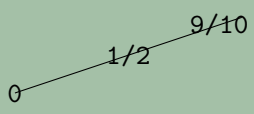
very near end 等于 pos=0.875

at start 等于 pos=0

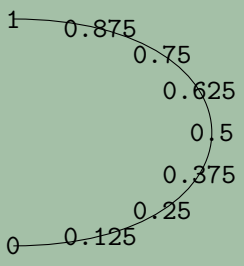
at end 等于 pos=1

pos=0 指示把 node 标签在起点，pos=1 指示把 node 标签在终点

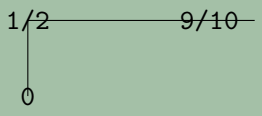
```
\tikz \draw (0,0) -- (3,1)
      node[pos=0]{0} node[pos=0.5]{1/2} node[pos=0.9]{9/10};
```




```
\tikz \draw (0,0) .. controls +(right:3.5cm)
            and +(right:3.5cm) .. (0,3)
      node foreach \p in {0,0.125,...,1} [pos=\p]{\p};
```

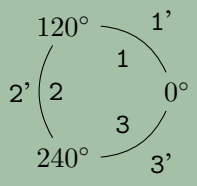


```
\tikz \draw (0,0) |- (3,1)
      node[pos=0]{0} node[pos=0.5]{1/2} node[pos=0.9]{9/10};
```



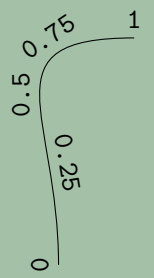
选项 auto 指示 node 标签在线的左侧或右侧（默认是 scope 的设置），auto=left 指示 node 标签在线的左侧，auto=right 指示 node 标签在线的右侧。选项 swap 指示 node 标签放在与 auto 相反的一侧，英文的撇号（一个单引号）' 是 swap 的简写。

```
\begin{tikzpicture}[auto,bend right]
\node (a) at (0:1) {$0^\circ$};
\node (b) at (120:1) {$120^\circ$};
\node (c) at (240:1) {$240^\circ$};
\draw (a) to node {1} node [swap] {1' } (b)
     (b) to node {2} node [swap] {2' } (c)
     (c) to node {3} node [swap] {3' } (a);
\end{tikzpicture}
```



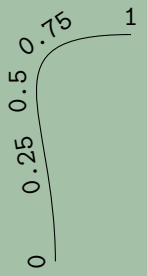
选项 sloped 使得 node 标签沿着路径倾斜

```
\tikz \draw (0,0) .. controls +(up:2cm)
          and +(left:2cm) .. (1,3)
          node foreach \p in {0,0.25,...,1}
              [sloped,above,pos=\p]{\p};
```



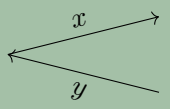
选项 allow upside down=true 或 allow upside down 使得 node 标签总在路径线的一侧，而不是只在路径线的凸的一侧。

```
\tikz [allow upside down]
\draw (0,0) .. controls +(up:2cm)
     and +(left:2cm) .. (1,3)
     node foreach \p in {0,0.25,...,1}
        [sloped,above,pos=\p]{\p};
```



选项 midway 将 node 标签放在路径中间，并可以与 right，left，above，below 等方位一起使用

```
\begin{tikzpicture}[->]
\draw (0,0) -- (2,0.5) node[midway,sloped,above] {$x$};
\draw (2,-.5) -- (0,0) node[midway,sloped,below] {$y$};
\end{tikzpicture}
```



7.14 给 node 加 label 标签，label 的用法

给 node 加标签可用两种方法，方法一，用\node，这个方法已如前所述。方法二，用 label 选项。

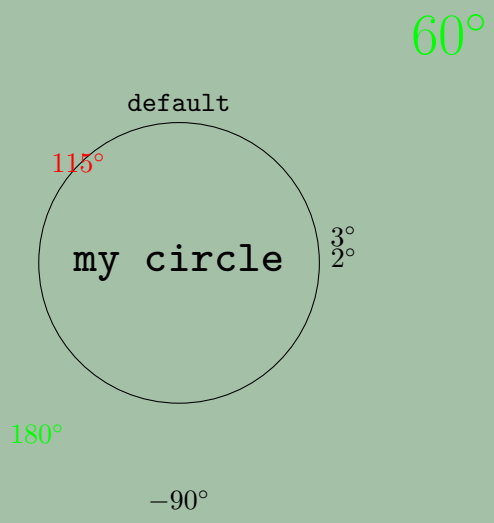
label 的句法

```
label=[options]<angle>:<text>
```

其中 angle 是标签所指向的对象的方位，可以是角度（方向角，有正负），也可以是 above，below，left，right，above left 等等。text 是标签内容。

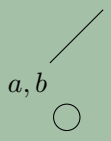
注意下面代码中的 label 的选项要用花括号括起来。

```
\tikz
\node [circle, draw,scale=2,
        label=default,
        label={[green]60:\Huge $60^\circ$},
        label=below:$-90^\circ$,
        label=3:$3^\circ$,
        label=2:$2^\circ$,
        label={[green,below]180:$180^\circ$},
        label={[red,centered]135:$115^\circ$}]
    {\tiny my circle};
```



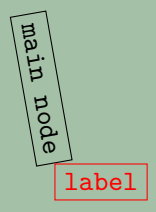
可以给标签命名，加以引用

```
\begin{tikzpicture}
\node [circle,draw,label={[name=label node]
      above left:$a,b$}] {};
\draw (label node) -- +(1,1);
\end{tikzpicture}
```



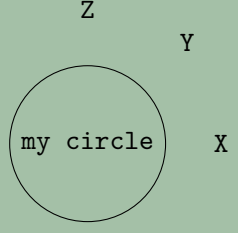
可用环境选项 every label/.style={} 设置各个 label 的样式

```
\tikz [rotate=-80,every label/.style={draw,red}]
\node [transform shape,rectangle,draw,label=right:label] {main node};
```

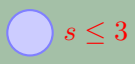


选项 label distance 设置标签与标签所指向的 node 对象之间的距离

```
\tikz[label distance=5mm]
\node [circle,draw,label=right:X,
      label=above right:Y,
      label=above:Z] {my circle};
```



```
\begin{tikzpicture}
    [place/.style={circle,draw=blue!50,fill=blue!20,thick,
    inner sep=0pt,minimum size=6mm}]
\node[place, label=right:$s\le3$] (waiting) at (0,2) {};
\end{tikzpicture}
```

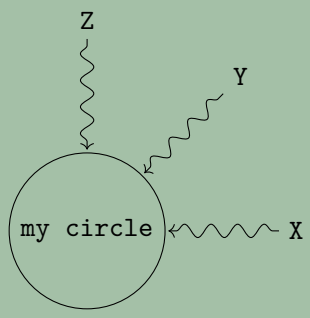


7.15 大头针 pin 标签

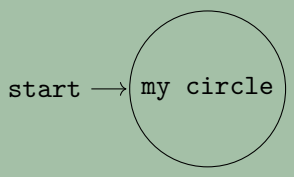
pin 标签的用法与 label 类似，只是多了边的设置。

```
every pin edge/.style={ }
pin edge=<options>
pin distance=<distance> (初始值 3ex) “大头”与指向的 node 对象之间的距离
```

```
\tikz [pin distance=15mm,
      every pin edge/.style={<-,shorten <=1pt,decorate,
       decoration={snake,pre length=4pt}}]
\node [circle,draw,pin=right:X,
      pin=above right:Y,
      pin=above:Z] {my circle};
```




```
\tikz [every pin edge/.style={},
       initial/.style={pin={[pin distance=5mm,
                 pin edge={<-,shorten <=1pt}]
      left:start}}]
\node [circle,draw,initial] {my circle};
```

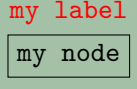


7.16 用引用选项 quotes 给 node 加标签

首先调用 quotes 程序库 \usetikzlibrary{quotes}。

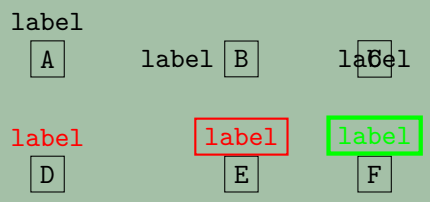
在 \node 的选项中，用双引号引起的字符串（不含某些特殊符号）会被作为标签引用。下面两句等效

```
\tikz \node ["my label" red, draw] {my node};
\tikz \node [label={[red]my label}, draw] {my node};
```

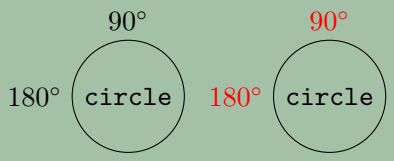


显然双引号简捷一些。

```
\begin{tikzpicture} % 调用 matrix 程序库
\matrix [row sep=5mm,column sep=5mm] {
\node [draw, "label"] {A}; &
\node [draw, "label" left] {B}; &
\node [draw, "label" centered] {C}; \\
\node [draw, "label" color=red] {D}; &
\node [draw, "label" {red,draw,thick}] {E} ;&
\node [draw, "label" {green,draw,ultra thick}] {F};\\ };
\end{tikzpicture}
```

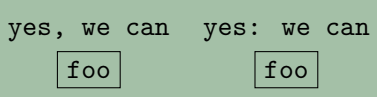



```
\tikz \node ["90:$90^\circ$", "left:$180^\circ$",
          circle, draw] {circle};
\tikz [every label quotes/.style=red]
     \node ["90:$90^\circ$", "left:$180^\circ$",
          circle, draw] {circle};
```



如果引号内有逗号或冒号，要用花括号括起来。

```
\tikz \node ["{yes, we can}", draw] {foo};
\tikz \node ["yes{:} we can", draw] {foo};
```



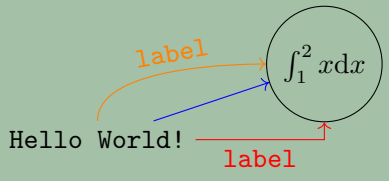
8 在 node 之间画线

8.1 用符号 --，..，-| 画线

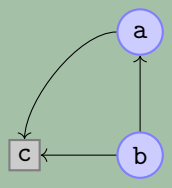
可以在坐标、node 之间画连线。基本线条样式有 3 种: 用 -- 表示的直线段，用 .. 表示的曲线，用 -| 或 |-表示的直角连线。

连线的起止点可以是坐标，node 的锚位置，node 的名称，带方位后缀的 node 的名称。（参照 node 的位置）

```
\begin{tikzpicture}
\path (0,0) node (x) {Hello World!}
(3,1) node[circle,draw](y) {$\int_1^2 x \mathrm d x$};
\draw[->,blue] (x) -- (y);
\draw[->,red] (x) -| node[near start,below] {label} (y);
\draw[->,orange] (x) .. controls +(up:1cm) and +(left:1cm)
                 .. node[above,sloped] {label} (y);
\end{tikzpicture}
```



```
\begin{tikzpicture}
     [place/.style={circle,draw=blue!50,fill=blue!20,thick,
      inner sep=0pt,minimum size=6mm},
      transition/.style={rectangle,draw=black!50,fill=black!20,
      thick,inner sep=0pt,minimum size=4mm}]
\node[place] (a) {a};
\node[place] (b) [below=of a] {b};
\node[transition] (c) [left=of b] {c};
\draw [->] (b.north) -- (a.south);
\draw [->] (b.west) -- (c.east);
\draw [->] (a.west) .. controls +(left:5mm) and +(up:5mm)
          .. (c.north);
\end{tikzpicture}
```



8.2 算子 to

8.2.1 出入角度设置

也可以用算子 to 代替连接线符号 -- 或 .. 画线

to[options] <nodes> <coordinate> <cycle>

to 可带有选项 out=，in=，bend left=，bend right= 等来指示连线起止点的位置。

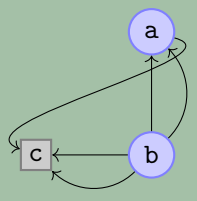
out=<angle> 以起点为原点建立直角坐标系，以x轴正方向为初始边，逆时针方向为正，这样规定下的 <angle> 是起点处的出发方向角;
in=<angle> 的意思与之一样，只是以终点为原点。

bend left=<angle> 以出发点为原点，起点至终点的位移向量为初始边方向，逆时针方向（左偏）为正，这样规定下的 <angle> 是起点处的出发方向角；而在终点处的进入方向是 -<angle>。
bend right=<angle>的意思与之类似，只是角的方向相反。如果不指定角度，它的默认值是前面最近出现的角度。

bend angle=<angle> 为bend left和bend right设置 <angle>。

出入角度的设置会影响边的形态，如下例所示

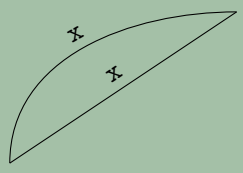
```
\begin{tikzpicture}
    [place/.style={circle,draw=blue!50,fill=blue!20,thick,
     inner sep=0pt,minimum size=6mm},
     transition/.style={rectangle,draw=black!50,fill=black!20,
     thick,inner sep=0pt,minimum size=4mm}]
\node[place] (a) {a};
\node[place] (b) [below=of a] {b};
\node[transition] (c) [left=of b] {c};
\draw [->] (b) to (a);
\draw [->] (b) to (c);
\draw [->] (a) to [out=-15,in=160] (c);
\draw [->] (b) to [bend right=45] (a);
\draw [->] (b) to [bend left=45] (c);
\end{tikzpicture}
```



8.2.2 加标签

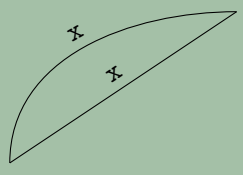
可以在 to 后接 node 给边加 node 标签

```
\begin{tikzpicture}
\draw (0,0) to node [sloped,above] {x} (3,2);
\draw (0,0) to[out=90,in=180] node [sloped,above] {x} (3,2);
\end{tikzpicture}
```



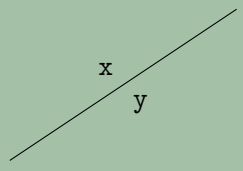
可以给 to 加 edge node=<node specification> 选项给边加 node 标签，注意下面用花括号把带方括号的内容括起来

```
\begin{tikzpicture}
\draw (0,0) to [edge node={node [sloped,above] {x}}] (3,2);
\draw (0,0) to [out=90,in=180,
edge node={node [sloped,above] {x}}] (3,2);
\end{tikzpicture}
```



可以给 to 加 edge label=<text> 或 edge label'=<text> 选项给边加标签，其中的撇号表示转换标签的位置

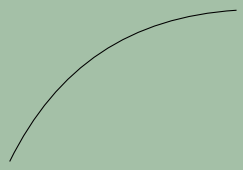
```
\tikz \draw (0,0) to [edge label=x, edge label'=y] (3,2);
```



8.2.3 设置每条边的样式

可以用 every to/.style 设置每条边的样式

```
\tikz[every to/.style={bend left}]
\draw (0,0) to (3,2);
```



8.3 路径算子 edge

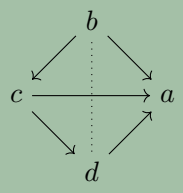
edge 句法格式

```
edge [options] <nodes> <coordinate> <cycle>
```

8.3.1 按一对多方式画线

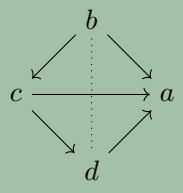
算子 to 是一对一地画线，而 edge 可以一对多地画线。注意下图中从 d 到 a，b，c 有 3 个 edge，只在结尾用一个分号。

```
\begin{tikzpicture}
\node (a) at (0:1) {$a$};
\node (b) at (90:1) {$b$} edge [->] (a);
\node (c) at (180:1) {$c$} edge [->] (a)
                           edge [<-] (b);
\node (d) at (270:1) {$d$} edge [->] (a)
                           edge [dotted] (b)
                           edge [<-] (c);
\end{tikzpicture}
```



上图也可以如下作出

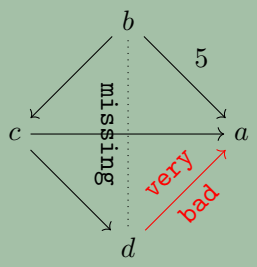
```
\begin{tikzpicture}
\node foreach \name/\angle in {a/0,b/90,c/180,d/270}
(\name) at (\angle:1) {$\name$};
\path[->] (b) edge (a)
              edge (c)
              edge [-,dotted] (d)
          (c) edge (a)
              edge (d)
          (d) edge (a);
\end{tikzpicture}
```



8.3.2 加 node 标签

也可以为 edge 加 node 标签

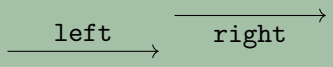
```
\begin{tikzpicture}
\node foreach \name/\angle in {a/0,b/90,c/180,d/270}
          (\name) at (\angle:1.5) {$\name$};
\path[->] (b) edge node[above right] {$5$} (a)
            edge (c)
            edge [-,dotted] node[below,sloped] {missing} (d)
        (c) edge (a)
            edge (d)
        (d) edge [red] node[above,sloped] {very}
                       node[below,sloped] {bad} (a);
\end{tikzpicture}
```



8.3.3 加 quotes 标签

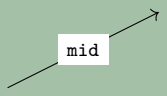
调用 quotes 程序库，也可以为 edge 使用 every edge quotes/.style={} 和 quotes 选项来加标签。

```
\tikz \draw (0,0) edge ["left", ->] (2,0);
\tikz [every edge quotes/.style={auto=right}]
     \draw (0,0) edge ["right", ->] (2,0);
```



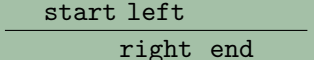
注意，如果在 every edge quotes/.style={} 中没有 auto 选项，则引用的字符串位于线上，而不是在线旁边。

```
\tikz [every edge quotes/.style={fill=white,font=\footnotesize}]
\draw (0,0) edge ["mid", ->] (2,1);
```



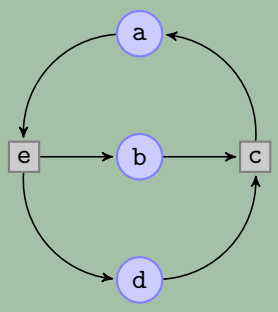
在双引号之后加撇号（一个单引号），表示 swap，便于转换字符串标签的侧位。

```
\tikz
\draw (0,0) edge ["left", "right"',
                "start" near start,
                 "end"' near end] (4,0);
```



8.4 重定义样式 pre 和 post

```
\begin{tikzpicture}
     [place/.style={circle,draw=blue!50,fill=blue!20,thick,
      inner sep=0pt,minimum size=6mm},
      transition/.style={rectangle,draw=black!50,fill=black!20,
      thick,inner sep=0pt,minimum size=4mm}]
     [bend angle=45,
      pre/.style={<-,shorten <=1pt,>=stealth',semithick},
      post/.style={->,shorten >=1pt,>=stealth',semithick}]
\node[place] (a) {a};
\node[place] (b) [below=of waiting] {b};
\node[place] (d) [below=of critical] {d};
\node[transition] (c) [right=of critical] {c}
                edge [pre] (b)
                edge [post,bend right] (a)
                edge [pre, bend left] (d);
\node[transition] (e) [left=of critical] {e}
                edge [post] (b)
                edge [pre, bend left] (a)
                edge [post,bend right] (d);
\end{tikzpicture}
```



9 作图层

不同环境的图做在不同的层上，如果后面环境的图与前面的有重叠，则重叠部分显示为后做的图，即后来的层覆盖前面的层。可以声明（或者说定义）带名称的层，并规定它们的叠放次序，例如

```
\pgfdeclarelayer{name1}
\pgfdeclarelayer{name2}
\pgfdeclarelayer{name3}
\pgfsetlayers{name1,name2,name3}
```

这样就声明了三个层 name1，name2，name3，并按照输入次序，规定它们自下而上叠放，最上是 name3 层。然后可以在不同的层上作图，用下面的环境

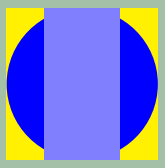
```
\begin{pgfonlayer}{<layer name>}
<environment contents>
\end{pgfonlayer}
```

该环境可以是其他环境的子环境。

```
\pgfdeclarelayer{background layer}
\pgfdeclarelayer{foreground layer}
\pgfsetlayers{background layer,main,foreground layer}
\begin{tikzpicture}
% On main layer:
\fill[blue] (0,0) circle (1cm);
\begin{pgfonlayer}{background layer}
\fill[yellow] (-1,-1) rectangle (1,1);
\end{pgfonlayer}
\begin{pgfonlayer}{foreground layer}
\node[white] {foreground};
\end{pgfonlayer}
\begin{pgfonlayer}{background layer}
\fill[black] (-.8,-.8) rectangle (.8,.8);
\end{pgfonlayer}
% On main layer again:
\fill[blue!50] (-.5,-1) rectangle (.5,1);
\end{tikzpicture}
```

背景层选项 on background layer 只能用于 scope 环境或者用在命令\scoped 后面，是主层之下的层。

```
\begin{tikzpicture}
% On main layer:
\fill[blue] (0,0) circle (1cm);
\begin{scope}[on background layer={color=yellow}]
\fill (-1,-1) rectangle (1,1);
\end{scope}
\begin{scope}[on background layer]
\fill[black] (-.8,-.8) rectangle (.8,.8);
\end{scope}
% On main layer again:
\fill[blue!50] (-.5,-1) rectangle (.5,1);
\end{tikzpicture}
```



10 node 的 fit 选项

node 的 fit 选项需要调用 fit 程序库，在导言区设置\usetikzlibrary{fit}

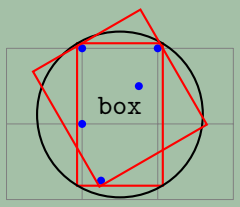
句法

```
fit=coordinates 或 nodes
```

这种语句生成一个盒子，例如 fit=(a) (b) (c) (d) (e)，等号右侧各项是坐标或 node 的名称，各项之间不能加逗号，用空格分隔，生成的盒子将列举的各项包围在内。各项与盒子边界的距离用选项“inner sep= 距离”来设置。

选项 rotate fit=angle 设置盒子的旋转角度。

```
\begin{tikzpicture}[inner sep=0pt,thick,
dot/.style={fill=blue,circle,minimum size=3pt}]
\draw[help lines] (0,0) grid (3,2);
\node[dot] (a) at (1,1) {};
\node[dot] (b) at (2,2) {};
\node[dot] (c) at (1,2) {};
\node[dot] (d) at (1.25,0.25) {};
\node[dot] (e) at (1.75,1.5) {};
\node[draw=red, fit=(a) (b) (c) (d) (e)] {box};
\node[draw,circle,fit=(a) (b) (c) (d) (e)] {};
\node[draw=red, rotate fit=30, fit=(a) (b) (c) (d) (e)] {};
\end{tikzpicture}
```



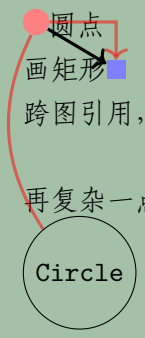
11 在多个图形之间引用 node

11.1 remember picture 选项和 overlay 选项

可以在某个环境或图形中引用其他环境或图形中的 node，跨图作图。给包含被引用 node 的图形加 remember picture 选项；给包含被引用 node 的路径或图形加 overlay 选项（该选项确保图形不会改变尺寸）。然后就可以在其他环境图形中引用该路径或图形中的 node。可能需多次编译排版才得到需要的图形。

观察下面的代码的作用

```
{\noindent \tikz[remember picture] \node[circle,fill=red!50] (n1) {};圆点\\
画矩形\tikz[remember picture] \node[fill=blue!50] (n2) {};\\
跨图引用，在圆点和矩形之间画箭头\\
\begin{tikzpicture}[remember picture,overlay]
\draw[->,very thick] (n1) -- (n2);
\end{tikzpicture}\\
再复杂一点\\
\begin{tikzpicture}[remember picture]
\node (c) [circle,draw] {Big circle};
\draw [overlay,->,very thick,red,opacity=.5]
(c) to[bend left] (n1) (n1) -| (n2);
\end{tikzpicture}}
```



11.2 Current Page Node -Absolute Positioning

current page 是一个特殊的预定义 node,这个选项把当前页作为一个 node,所以 current page.south west 是当前页的左下角位置，current page.center 是当前页的中心。下面的例子引用 current page 为页面添加特殊效果。

在当前页的左下角添加文字

```
\begin{tikzpicture}[remember picture,overlay]
\node [xshift=1cm,yshift=1cm] at (current page.south west)
[text width=7cm,fill=red!20,rounded corners,above right]
{
This is an absolutely positioned text in the
lower left corner. No shipout-hackery is used.
};
\end{tikzpicture}
```

在当前页的中心画一个不透明度为 0.25 的圆

```
\begin{tikzpicture}[remember picture,overlay]
\draw [line width=1mm,opacity=.25,red]
(current page.center) circle (3cm);
\end{tikzpicture}
```

在当前页的中心添加不透明度为 0.2 的文字

```
\begin{tikzpicture}[remember picture,overlay]
\node [rotate=60,scale=10,text=green,text opacity=0.2]
at (current page.center) {Example};
\end{tikzpicture}
```

This is an absolutely positioned text in the lower left corner. No shipout-hackery is used.

12 关于平面几何作图的几个工具

先调用程序库\usetikzlibrary[calc,intersections,through,backgrounds]

12.1 coordinate 算子

coordinate 算子用于声明坐标，可用以下句法得到 coordinate 对象

```
\coordinate
\path . . . coordinate [options] (name) at (coordinate) . . . ;
\node [shape=coordinate] [options] (name) at (coordinate) {};
```

画线段并给端点命名

```
\begin{tikzpicture}
\coordinate [label=left:\textcolor{blue}{$A$}] (A) at (0,0);
\coordinate [label=right:\textcolor{blue}{$B$}] (B) at (1.25,0.25);
\draw[blue] (A) -- (B);
\end{tikzpicture}
```

12.2 rand

函数 rand 生成-1 到 1 之间的随机数，用它画随机点

```
\coordinate [...] (A) at (0+0.1*rand,0+0.1*rand);
\coordinate [...] (B) at (1.25+0.1*rand,0.25+0.1*rand);
```

也可以写成

```
\coordinate [...] (A) at ($ (0,0) + .1*(rand,rand) $);
\coordinate [...] (B) at ($ (1.25,0.25) + .1*(rand,rand) $);
```

注意小数或分数与 rand 相乘要加 * 号，这样的乘积可以套嵌。这里两个美元符号表示计算，而不是 LaTeX 的数学模式。

12.3 let 赋值算子

可用 let 算子对表达式中的符号作定义

```
\path . . . let <assignment>,<assignment>,<assignment>. . . in . . . ;
```

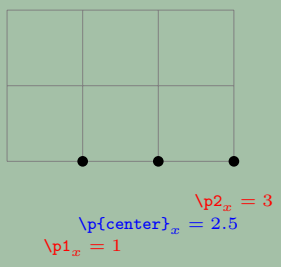
其中 in 后是表达式，表达式中的符号由 assignment 定义。assignment 的形式是“符号 = 数值或表达式”。在 assignment 中常用宏\p，\x，\y，\n。如果定义

```
let \p1=(1pt,1pt+2pt), \p2=(4,3), \p{name}=(0,5)...,
```

则\p1 就是 7 个字符“1pt,3pt”，要是加括号(\p1) 就得到点 (1pt,3pt)。对应的，\x1 就是(\p1) 的第一个坐标分量 1pt，\y1 就是(\p1) 的第二个坐标分量 3pt，\x2 就是 4，\y2 就是 3，\x{name} 就是 0，\y{name} 就是 5。注意，用 let 定义的点坐标都以 pt 为单位存在，即 LaTeX 符号$\x2$, $\y2$ 显示为带 pt 的数字。

观察下例，注意在 in 后各行有无分号，带分号的行结束 let 赋值。

```
\begin{tikzpicture}
\draw [help lines] (0,0) grid (3,2);
\fill
     let
         \p1 = (1,0),
         \p2 = (3,2),
         \p{center} = ($ (\p1) !.5! (\p2) $)
     in
          (\x1,0) node[below=0.9cm,red]{\scriptsize$\verb!\p1!_x=1$}
           circle [radius=2pt]
          (\x2,0) node[below=0.3cm,red]{\scriptsize$\verb!\p2!_x=3$}
           circle [radius=2pt]
          (\x{center},0) node[below=0.6cm,blue]{\scriptsize$\verb!\p{center}!_x=2.5$}
          circle [radius=2pt] ;
\end{tikzpicture}
```

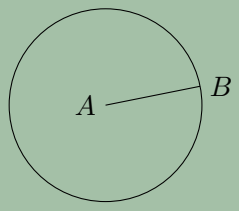


12.4 函数 veclen(x,y)

函数 veclen(x,y) 的值是 $\sqrt{x^2+y^2}$，即向量 (x,y) 的长度。下面的例子中，用函数 veclen(x,y) 来构造 let 算子的赋值过程。

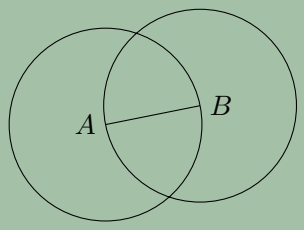
函数 veclen(x,y) 作为 circle 的参数

```
\begin{tikzpicture}
\coordinate [label=left:$A$] (A) at (0,0);
\coordinate [label=right:$B$] (B) at (1.25,0.25);
\draw (A) -- (B);
\draw (A) let
            \p1 = ($ (B) - (A) $)
        in
            circle ({veclen(\x1,\y1)});
\end{tikzpicture}
```



将函数 veclen(x,y) 的值转化为宏，再加以利用

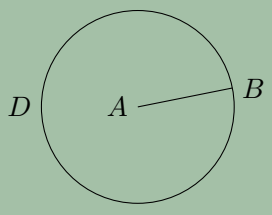
```
\begin{tikzpicture}
\coordinate [label=left:$A$] (A) at (0,0);
\coordinate [label=right:$B$] (B) at (1.25,0.25);
\draw (A) -- (B);
\draw let \p1 = ($ (B) - (A) $),
        \n{radius} = {veclen(\x1,\y1)}
     in
        (A) circle (\n{radius})
        (B) circle (\n{radius});
\end{tikzpicture}
```



12.5 circle through 选项

给 \node 加选项 circle through=<coordinate>,将以 node 指向的对象位置为圆心画过 coordinate 的圆，这需要调用 through 程序库。

```
\begin{tikzpicture}
\coordinate [label=left:$A$] (A) at (0,0);
\coordinate [label=right:$B$] (B) at (1.25,0.25);
\draw (A) -- (B);
\node [draw,circle through=(B),label=left:$D$] at (A) {};
\end{tikzpicture}
```



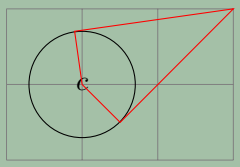
12.6 切线坐标系统

首先调用 cal 程序库。

tangent cs:node=<node> 指定切线切于<node>的边界，<node>的形状只能是 coordinate 或 circle
tangent cs:point=<point> 指定切线通过点<point>

tangent cs:solution=<number> 如果切点多于1个，指定需要被操作的切点

```
\begin{tikzpicture}
\draw[help lines] (0,0) grid (3,2);
\coordinate (a) at (3,2);
\node [circle,draw] (c) at (1,1) [minimum size=40pt] {$c$};
\draw[red] (a) -- (tangent cs:node=c,point={(a)},solution=1)
            -- (c.center) -- (tangent cs:node=c,point={(a)},
               solution=2) -- cycle;
\end{tikzpicture}
```

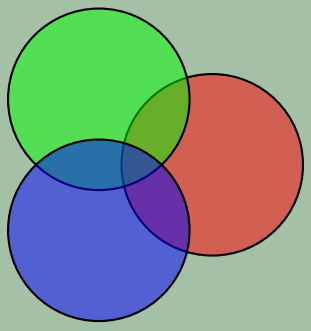


13 透明度设置

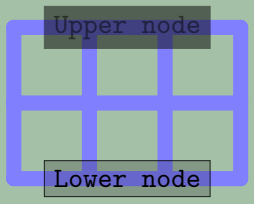
可用选项 opacity=<value> 设置不透明度。选项 opaque 指示完全不透明。例如 draw opacity=0.5，fill opacity=0.5，text opacity=1 分别设置路径、填充色、文字的不透明度。

```
\begin{tikzpicture}[line width=1ex]
\draw (0,0) -- (3,1);
\filldraw [fill=yellow!80!black,draw opacity=0.5]
        (1,0) rectangle (2,1);
\end{tikzpicture}
```

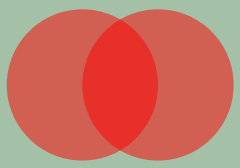
```
\begin{tikzpicture}[thick,fill opacity=0.5]
\filldraw[fill=red] (0:1cm) circle (12mm);
\filldraw[fill=green] (120:1cm) circle (12mm);
\filldraw[fill=blue] (-120:1cm) circle (12mm);
\end{tikzpicture}
```



```
\begin{tikzpicture}[every node/.style={fill,draw}]
\draw[line width=2mm,blue!50,line cap=round] (0,0) grid (3,2);
\node[opacity=0.5] at (1.5,2) {Upper node};
\node[draw opacity=0.8,fill opacity=0.2,text opacity=1]
     at (1.5,0) {Lower node};
\end{tikzpicture}
```



```
\begin{tikzpicture}[fill opacity=0.5]
\fill[red] (0,0) circle (1);
\fill[red] (1,0) circle (1);
\end{tikzpicture}
```



14 箭头

14.1 箭头样式选项的句法格式

```
arrows=<start arrow specification>-<end arrow specification>
```

此语句指定起点和终点的箭头样式（中间有短线）。如果未指定，例如 -> 未指定起点箭头样式，就不在起点画箭头。语句中的 arrows= 可以省略。

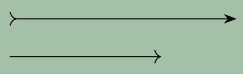
关于箭头的多种样式，参考《手册》16.5 Reference: Arrow Tips; 67.5 Arrow Shapes.

给 \draw 命令或含有 draw 选项的命令添加选项 -> 或 -< 可以在路径末端添加箭头；添加选项 <- 或 >- 可以在路径始端添加箭头；添加选项 <-> 可以在路径始末两端添加箭头。只能给有端点的路径加箭头。当未指定箭头样式时，使用 clip 命令时，tips=never 或 false 时，tips=on draw 但无 draw 命令时，\path 为空时，点在环路上时，都没有箭头。

注意 -Stealth, ->Stealth, -Stealth[] >, -Stealth[] > 合法。

-Stealth> 不合法。

```
\begin{tikzpicture}
\draw[->] (0,0) -- (2,0);
\draw[>-Stealth] (0,0.5) -- (3,0.5);
\end{tikzpicture}
```



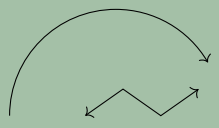
作为其他命令的选项的箭头样式自己也可以带选项，此时必须把箭头样式用花括号括起来

```
\draw [-{Stealth[length=5mm]}] ...
```

选项 tip 可以指示箭头是否画出

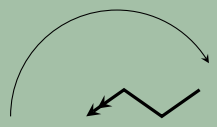
```
tips=<value> 默认 true，初值 on draw，其中 value 可以取值 true，proper，on draw，
on proper draw，never 或 false。
```

```
\begin{tikzpicture}
\draw [->] (0,0) arc [start angle=180, end angle=30,
                      radius=40pt];
\draw [<->] (1,0) -- (1.5cm,10pt) -- (2cm,0pt) -- (2.5cm,10pt);
\end{tikzpicture}
```

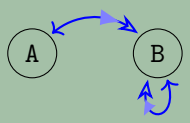


可以把箭头样式作为环境选项

```
\begin{tikzpicture}[>=stealth]
\draw [->] (0,0) arc [start angle=180, end angle=30, radius=40pt];
\draw [<<-,very thick] (1,0) -- (1.5cm,10pt) --
     (2cm,0pt) -- (2.5cm,10pt);
\end{tikzpicture}
```



```
\tikz {
\node [circle,draw] (A) {A};
\node [circle,draw] (B) [right=of A] {B};
\draw [draw = blue, thick,
     arrows={
             Computer Modern Rightarrow [sep]
             - Latex[blue!50,length=8pt,bend,line width=0pt]
             Stealth[length=8pt,open,bend,sep]}]
             (A) edge [bend left=45] (B)
             (B) edge [in=-110, out=-70,looseness=8] (B);
            }
```



14.2 箭头长度选项

```
length=<dimension>␣<line␣width␣factor>␣<outer␣factor>
```

注意这三个选项之间有空格。第一选项 dimension 是箭头基本长度，例如 5mm；第二选项是个（不带单位的纯数值）倍数因子 w，使得箭头尺寸在 dimension 的基础上再增加 “w 与线宽的乘积”；第三选项是在使用双线 double 选项时才有效，指的是双线的间距。

```
\tikz{
\draw [-{Stealth[length=5mm]}] (0,0) -- (2,0);
\draw [|<->|] (1.5,.4) -- node[above=1mm] {5mm} (2,.4);
}
```


```
double 指示画双线
double distance=<dimension> 设置双线的间距
```

```
\tikz \draw [line width=1pt, double distance=3pt,
arrows = {-Latex[length=0pt 3 0]}] (0,0) -- (1,0);
```

14.3 箭头宽度选项

```
width=<dimension>␣<line␣width␣factor>␣<outer␣factor>
```

与长度选项类似，注意这三个选项之间有空格。

```
width'=<dimension>␣<␣length␣factor>␣<line␣width␣factor>
```

此格式中，第一选项设置箭头宽度为 dimension；第二选项是个数值，指示在 dimension 基础上再加箭头长度的倍数；第三选项是数值，指示在前两项的基础上再加线宽的倍数。这个格式容易使得箭头宽度、长度、线宽成一定比例。例如 width'=0pt 0.5 设置箭头长宽比为 2.

```
\tikz
\draw [arrows = {-Latex[width'=0pt .5, length=15pt]}] (0,0) -- (1,0);
```


14.4 调节箭头尾部内凹的深度

```
inset=<dimension>␣<line␣width␣factor>␣<outer␣factor>
inset'=<dimension>␣<␣length␣factor>␣<line␣width␣factor>
```

注意格式中的空格，格式的意义与 width' 类似。

比较

```
\tikz \draw [arrows = {-Stealth[length=10pt, inset=5pt]}] (0,0) -- (1,0);
\tikz \draw [arrows = {-Stealth[length=10pt, inset=2pt]}] (0,0) -- (1,0);
```

14.5 调节箭头尖端的角度

```
angle=<angle>:<dimension>␣<line␣width␣factor>␣<outer␣factor>
angle'=<angle>
```

注意格式中的空格，格式的意义与 width' 类似。

比较

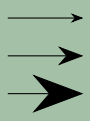
```
\tikz \draw [arrows = {-Stealth[inset=0pt, length=10pt, angle'=90]}]
          (0,0) -- (1,0);
\tikz \draw [arrows = {-Stealth[inset=0pt, length=10pt, angle'=30]}]
          (0,0) -- (1,0);
```

14.6 箭头依次变化尺寸

```
scale=<factor>
```

箭头带上这个选项，使得箭头尺寸（长，宽，内凹深度三方面）是默认箭头尺寸的 factor 倍，但不影响线宽。

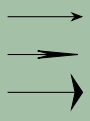
```
\tikz {
\draw [arrows = {-Stealth[]}] (0,1) -- (1,1);
\draw [arrows = {-Stealth[scale=2]}] (0,0.5) -- (1,0.5);
\draw [arrows = {-Stealth[scale=4]}] (0,0) -- (1,0);
}
```



```
scale length=<factor>
scale width=<factor>
```

这两个选项只针对箭头长度和宽度。

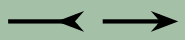
```
\tikz {
\draw [arrows = {-Stealth[]}] (0,1) -- (1,1);
\draw [arrows = {-Stealth[scale length=4]}] (0,0.5) -- (1,0.5);
\draw [arrows = {-Stealth[scale width=4]}] (0,0) -- (1,0);
}
```



14.7 箭头原地反向

选项 reverse 使得箭头原地反向

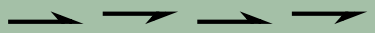
```
\tikz [ultra thick]
\draw [arrows = {-Stealth[reversed]}]
(0,0) -- (1,0);
\tikz [ultra thick]
\draw [arrows = {-Stealth[reversed, reversed]}]
(0,0) -- (1,0);
```



14.8 半箭头——鱼叉

选项 harpoon 显示半个箭头，等于选项 left；选项 swap 显示另一半箭头；而选项 right 等于 harpoon, swap 两个选项。

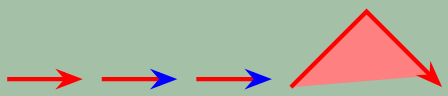
```
\tikz [ultra thick]
     \draw [arrows = {-Stealth[harpoon]}]
           (0,0) -- (1,0);
\tikz [ultra thick]
     \draw [arrows = {-Stealth[harpoon,swap]}]
           (0,0) -- (1,0);
\tikz [ultra thick]
     \draw [arrows = {-Stealth[left]}]
           (0,0) -- (1,0);
\tikz [ultra thick]
     \draw [arrows = {-Stealth[right]}]
           (0,0) -- (1,0);
```



14.9 箭头颜色

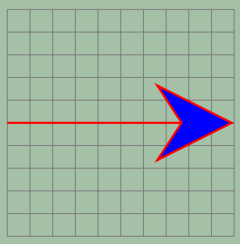
可以用\draw 命令画箭头边界线的颜色。

```
\tikz [ultra thick]
     \draw [red, arrows = {-Stealth}]
           (0,0) -- (1,0);
\tikz [ultra thick]
     \draw [red, arrows = {-Stealth[color=blue]}]
           (0,0) -- (1,0);
\tikz [ultra thick]
\draw [red, arrows = {-Stealth[blue]}]
      (0,0) -- (1,0);
\tikz [ultra thick]
     \draw [draw=red, fill=red!50,
           arrows = {-Stealth[length=10pt]}]
           (0,0) -- (1,1) -- (2,0);
```



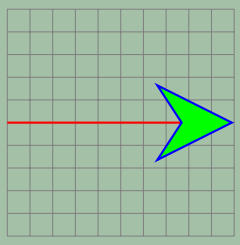
也可以用 fill=color 或 none 选项在箭头内部填充颜色。

```
\tikz {
      \draw [help lines,scale=3] (0,-.5) grid [step=1mm] (1,.5);
      \draw [thick, red, scale=3, arrows = {-Stealth[fill=blue,
            length=30pt,width'=0 1]}]
           (0,0) -- (1,0);
      }
```



下面图中，red 是整个\draw 的颜色，所以线是红色的；而选项 blue 修改了箭头颜色为蓝色，fill 又用绿色填充箭头内部。

```
\tikz {
\draw [help lines, scale=3] (0,-.5) grid [step=1mm] (1,.5);
\draw [thick, red, scale=3,
      arrows = {-Stealth[color=blue, fill=green,
       length=30pt,width'=0 1]}] (0,0) -- (1,0);
      }
```



14.10 箭头边界线宽

```
line␣width=<dimension>␣<line␣width␣factor>␣<outer␣factor>
line␣width'=<dimension>␣<length␣factor>
```

此格式意思与箭头长度、宽度选项类似。如果第一项 dimension 是 0，则只填充，无边界。

```
\tikz \draw [arrows = {-Latex[line width=5pt, fill=white,
            length=25pt,width'=0 1]}] (0,0) -- (2,0);
```

14.11 线冠，线结合，斜结限制

```
line cap=<type> 线冠类型
line join=<type> 线结合类型
miter limit=<factor> 斜结限制
```

线冠有 3 种类型，圆 round，方 rect，平（无线冠）butt

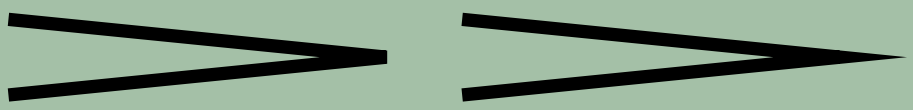
```
\begin{tikzpicture}
\begin{scope}[line width=15pt]
\draw[line cap=rect] (0,0 ) -- (1,0);
\draw[line cap=butt] (0,.5) -- (1,.5);
\draw[line cap=round] (0,1 ) -- (1,1);
\end{scope}
\draw[white,line width=2pt]
     (0,0 ) -- (1,0) (0,.5) -- (1,.5) (0,1 ) -- (1,1);
\end{tikzpicture}
```

线结合部有 3 个样式，圆 round，平 bevel，斜 miter

```
\begin{tikzpicture}[line width=10pt]
\draw[line join=round] (0,0) -- ++(.5,1) -- ++(.5,-1);
\draw[line join=bevel] (1.25,0) -- ++(.5,1) -- ++(.5,-1);
\draw[line join=miter] (2.5,0) -- ++(.5,1) -- ++(.5,-1);
\useasboundingbox (0,1.5); % enlarge bounding box
\end{tikzpicture}
```

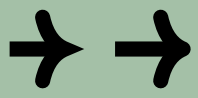
斜结限制 miter limit=<factor> 调节两线交接部的尖锐程度。

```
\begin{tikzpicture}[line width=5pt]
\draw (0,0) -- ++(5,.5) -- ++(-5,.5);
\draw[miter limit=25] (6,0) -- ++(5,.5) -- ++(-5,.5);
\useasboundingbox (14,0); % enlarge bounding box
\end{tikzpicture}
```



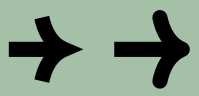
线冠选项，线结合选项，斜结限制也适用于箭头，可以作为箭头的选项。

```
\tikz [line width=2mm]
\draw [arrows = {-Computer Modern Rightarrow[line join=miter]}]
      (0,0) -- (1,0);
\tikz [line width=2mm]
\draw [arrows = {-Computer Modern Rightarrow[line cap=round]}]
     (0,0) -- (1,0);
```



也可以略去 line cap，line join，改用样式 round，sharp

```
\tikz [line width=2mm]
\draw [arrows = {-Computer Modern Rightarrow[sharp]}]
      (0,0) -- (1,0);
\tikz [line width=2mm]
\draw [arrows = {-Computer Modern Rightarrow[round]}]
      (0,0) -- (1,0);
```



14.12 弯曲箭头

箭头带 bend 选项会使曲线上的箭头随着曲线的弯曲而弯曲。

```
\begin{tikzpicture}
\draw [red,line width=1mm,-{Stealth[bend,round,length=30pt]}]
(0,-.5) .. controls (1,-.5) and (0.5,0) .. (1.5,0);
\end{tikzpicture}
```

14.13 串联箭头及其间距

->>> 是短线 - 和 3 个 > 的组合，与-[]>>> 和 ->[]>[]>[] 等效，生成 3 个紧紧相邻的箭头。

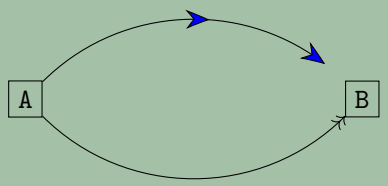
选项

```
sep=<dimension>␣<line␣width␣factor>␣<outer␣factor>
```

用在两个对象之间，设置二者的间距，默认sep=0.88pt .3 1（注意这个默认值是三个数量格式的）

```
\tikz {
\draw [very thick,-{>[sep=1pt]>[sep= 6pt]>}]
      (0,1.0) -- (1,1.0);
\draw [very thick,-{>[sep=6pt]>[sep=-2pt]>}]
      (0,0.5) -- (1,0.5);
\draw [very thick,-{> >[sep] >}]
       (0,0.0) -- (1,0.0);
      }
```

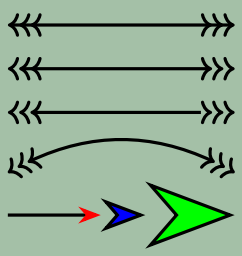
```
\tikz {
\node [draw] (A) {A};
\node [draw,node distance=4cm] (B) [right=of A] {B};
\draw [>={Stealth[scale=2,fill=blue,bend]}]
     [-{> [sep=40pt] >[sep=10pt]}]
      (A) to [bend left=45] (B);
\draw [- > >] (A) to [bend right=45] (B);
       }
```



14.14 间断串联箭头

可以在某个箭头前或后加一个点号，使得点号之后的路径不显示，造成多个箭头不连续的效果。

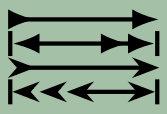
```
\tikz [very thick]
     \draw [<<<->>>] (0,0) -- (3,0);\\
\tikz [very thick]
     \draw [<.<<->.>>] (0,0) -- (3,0);\\
\tikz [very thick]
     \draw [<<.<-.>>>] (0,0) -- (3,0);\\
\tikz [very thick]
     \draw [<<.<->.>>] (0,0) to [bend left] (3,0);\\
\tikz [very thick]
     \draw [-{Stealth[red] . Stealth[scale=2,fill=blue]
           Stealth[scale=4,fill=green]}] (0,0) -- (3,0);
```



14.15 指示箭头的界限

将竖线符号 | 放在箭头符号一旁，可以给箭头指示界限。

```
\begin{tikzpicture}[scale=2,ultra thick]
\begin{scope}[>=Latex]
\draw[>->] (0pt,3ex) -- (1cm,3ex);
\draw[|<->>|] (0pt,2ex) -- (1cm,2ex);
\end{scope}
\begin{scope}[>=Stealth]
\draw[>->] (0pt,1ex) -- (1cm,1ex);
\draw[|<<.<->|] (0pt,0ex) -- (1cm,0ex);
\end{scope}
\end{tikzpicture}
```



15 数学程序库

在导言区调用程序库 `\usetikzlibrary{math,calc}`。

pgf 和 TikZ 都使用 pgf 数学引擎（这个引擎用纯 TEX 来计算，与专业的数学软件相比，计算的速度和精度有限）来执行数学表达式，但是所用的语法非常复杂。数学程序库会利用 pgf 数学引擎，但使用相对便捷的句法。可以调用 fpu 程序库来提高计算精度。数学程序库在执行关于坐标的运算时，会调用 cal 程序库。

数学程序库用在命令 `\tikzmath{<statements>}` 之中，其中 `<statements>` 是一系列执行数学运算的语句，每个语句必须以分号结束。

关键词 `evaluate=<statements>`，等效于 `\tikzmath{<statements>}`。

命令 evaluate=<statements> 和语句 \tikzmath{<statements>} 中的 <statements> 由一些数学程序库定义的关键词和表达式构成，注意每个关键词之后必须带一个空格，不能有空行。

15.1 用等号 = 赋值

可以用等号 = 给宏赋值，例如

```
\newcount\mycount
\newdimen\mydimen
\tikzmath{
        \a = 4*5+6;
        \b = sin(30)*4;
        \mycount = log10(2048) / log10(2);
        \mydimen = 15^2;
        }
\a, \b, \the\mycount, \the\mydimen
```

26.0, 2.0, 11, 225.0pt

被赋值的宏可以有数字后缀，作用类似数学下标

```
\tikzmath{
        \x1 = 3+4; \x2 = 30+40; \x3 = 300+400;
        }
\x1, \x2, \x3
```

7.0, 70.0, 700.0

被赋值的宏也可以有非数字后缀，只需把后缀用花括号括起来

```
\tikzmath{
        \c{air} = 340; \c{water} = 1435; \c{steel} = 6100;
        }
\foreach \medium in {air,steel}
     {The speed of sound in \medium\ is \c{\medium} m/s. }
```

The speed of sound in air is 340 m/s. The speed of sound in steel is 6100 m/s.

15.2 let 赋值

句法

```
let <variable> = <expression>;
```

```
\tikzmath{
        let \x = (5*4)+1;
        let \c1 = blue;
        }
\x, “\c1”
```

(5*4)+1, “blue”

15.3 数值类型

默认按照十进制小数做计算。也可以声明数值类型。

```
integer <variable>, <additional variables>; 声明所列举的变量是整数
int <variable>, <additional variables>; 上一语句的简写
real <variable>, <additional variables>; 声明所列举的变量是实数
```

```
\tikzmath{
         integer \x, \y, \z;
         \x = 4*5+6;
         \y = sin(30)*4;
         \z = log10(512) / log10(2);
         print {$x=\x$, $y=\y$, $z=\z$};
        }
```

$x = 26$, $y = 2$, $z = 9$

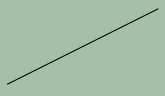
15.4 给 coordinate 变量赋值

可以把 (2cm,3pt),(my node.east),或者坐标的运算表达式(无需用 $...$ 符号)赋予某个 coordinate 变量(宏), 并可以在 tikzpicture 环境中引用这个宏。

句法

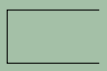
```
coordinate <variable>, <additional variables>;
```

```
\tikzmath{
        coordinate \c, \d;
        \c = (-1,2)+(1,-1);
        \d = (4,1)-(2,-1);
        }
\tikz\draw (\c) -- (\d);
```



坐标分量可用如下方式引用

```
\tikzmath{
         coordinate \c;
         \c1 = (30:20pt); % 加后缀
         \c2 = (210:20pt); % 加后缀
        }
\tikz\draw (\cx1,\cy1) -- (\cx2,\cy1) -- (\cx2,\cy2) -- (\cx1,\cy2);
```



注意，下面的坐标赋值语句都不能正常编译

```
\tikzmath{
coordinate \a,\b,\c;
           \a = (11,9);
           \b = (4,7);
           \c = \a - \b;
}
```

```
\tikzmath{
coordinate \a,\b;
           \a = (11,9);
           \b = (4,7);
let \c = \a - \b;
}
```

下面的坐标赋值语句则正常

```
\tikzmath{
real \m,\n;
\m=3.7;
\n=\m^3;
coordinate \a;
         \a = (\m,\n);}
```

15.5 for 循环语句

句法

```
for <variable> in {<list>}{<expressions>};
```

注意，(1) 使用花括号; (2) <list> 中不能有坐标; (3) 一个 for 后只能有一个变量; (4) for 引起的赋值的有效范围超出循环体。

```
\tikzmath{
        int \x, \y;
        \y = 0;
        for \x1 in {1,...,5}{
           for \x2 in {10,20,...,50}{
               \y = \y+\x1*\x2;
                                  };
                           };
        }
$x_1=\x1, x_2=\x2, y=\y$
```

$x_1 = 5, x_2 = 50, y = 2250$

15.6 if 条件语句

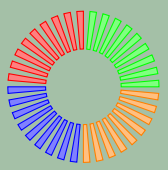
语句结构

```
if <condition> then {<if-non-zero-statements>};

if <condition> then {<if-non-zero-statements>}
            else {<if-non-zero-statements>};
```

例如

```
\begin{tikzpicture}
\tikzmath{
        int \x;
        for \k in {0,10,...,350}{
            if \k>260 then { let \c = orange; } else {
               if \k>170 then { let \c = blue; } else {
                 if \k>80 then { let \c = red; } else {
                    let \c = green; }; }; };
{\path [fill=\c!50, draw=\c] (\k:0.5cm) -- (\k:1cm) --
(\k+5:1cm) -- (\k+5:0.5cm) -- cycle;};
                                 };
        }
\end{tikzpicture}
```



15.7 声明函数

句法

```
function <name>(<arguments>) { <definition> };
```

其中的 name 可以是任何当前环境中尚未定义的名称；如果没有变量，可以不用圆括号；在函数定义中，必须用 return <expression>; 结尾。

```
\tikzmath{
        function product(\x,\y) {return \x*\y;};
        int \i, \i, \k;
        \i = random(1,10);
        \j = random(20, 40);
        \k = product(\i, \j);
        print { $\i\times \j = \k$ };
         }
```

$4 \times 39 = 156$

15.8 输出结果

按前面的例子，在 \tikzmath{ } 之后，列出表达式（多个表达式之间用逗号分隔，最后加分号），编译后即可得到表达式的执行结果。

也可以用 print 算子来输出结果，注意 print 算子用在 \tikzmath{ } 之内。

句法

```
print {<code>};
```

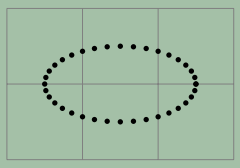
```
\tikzmath{
         int \x, \y, \z;
         \x = random(2, 5);
         for \y in {0,...,6}{
            \z = \x^\y;
            print {$\x^\y=\z$, };
                          };
         }
```

$2^0 = 1,\quad 2^1 = 2,\quad 2^2 = 4,\quad 2^3 = 8,\quad 2^4 = 16,\quad 2^5 = 32,\quad 2^6 = 64,$

15.9 注意赋值的有效范围

所有赋值在整个分组内有效，注意用环境或者花括号分组限制赋值的有效范围。

```
\begin{tikzpicture}
\draw [help lines] grid (3,2);
\tikzmath{
         coordinate \c;
         for \x in {0,10,...,360}{
            \c = (1.5cm, 1cm) + (\x:1cm and 0.5cm);
            { \fill (\c) circle [radius=1pt]; };
                              };
        }
\end{tikzpicture}
```



15.10 基本数学运算和函数

trig format=deg|rad (初始值 deg)
设置三角函数运算角度量制度，初始值是角度制

注意，有的运算符号与运算对象之间不能有空格，例如，不能在 5*3 之间插入空格。

x + y	加法
x - y	减法
- x	相反数
x * y	乘法
x / y	除法
x ^ y	乘方，幂运算
x!	阶乘
x r	把 x 转为弧度
x ? y : z	等价于 if x then y else z，若 x 非零即真
x==y	若 x 等于 y 则返回 1，否则返回 0
x > y	若 x 大于 y 则返回 1，否则返回 0
x < y	若 x 小于 y 则返回 1，否则返回 0
x = y!	若 x 不等于 y 则返回 1，否则返回 0
x >= y	若 x 大于等于 y 则返回 1，否则返回 0
x <= y	若 x 小于等于 y 则返回 1，否则返回 0
x && y	若 x 和 y 都是非零的则返回 1，否则返回 0，即合取，逻辑与
x || y	若 x 或 y 非零则返回 1，否则返回 0，即析取，逻辑或
!x	若 x 是零则返回 1，否则返回 0，即逻辑非
true	执行结果为 1
false	执行结果为 0
and(x,y)	x && y，逻辑与
or(x,y)	x || y，逻辑或
not(x)	!x，逻辑非

add(x,y)	x + y
subtract(x,y)	x - y
neg(x)	-x
multiply(x,y)	x * y
divide(x,y)	x / y
div(x,y)	按就近原则，对 x 除以 y 的商取整
sqrt(x)	x^(1/2)
pow(x,y)	x^y
e	自然对数底
exp(x)	e^x
ln(x)	以 e 为底的 x 的对数
log10(x)	以 10 为底的 x 的对数
log2(x)	以 2 为底的 x 的对数
abs(x)	x 的绝对值
mod(x,y)	x 除以 y 的余数
sign(x)	x 的符号
round(x)	对 x 四舍五入
floor(x)	按就近原则，对 x 取整
ceil(x)	对 x 向上取整
int(x)	返回 x 的整数部分
frac(x)	返回 x 的小数部分
real(x)	声明 x 是实数
gcd(x,y)	x 与 y 的最大公因子
isodd(x)	若 x 的整数部分是奇数则返回 1
iseven(x)	若 x 的整数部分是偶数则返回 1
isprime(x)	若 x 的整数部分是素数则返回 1
pi	圆周率 π
rad(x)	把 x 转为弧度
deg(x)	把 x 转为角度
sin(x)	角度 x 的正弦值
sin(x r)	弧度 x 的正弦值，例如 sin(pi/3 r)
cos(x)	角度 x 的余弦值
cos(x r)	弧度 x 的余弦值
tan(x)	角度 x 的正切值
tan(x r)	弧度 x 的正切值
sec(x)	角度 x 的正割
cosec(x)	角度 x 的余割
cot(x)	角度 x 的余切值
cot(x r)	弧度 x 的余切值
asin(x)	x 的反正弦角度值，在 $-90°$ 到 $90°$ 之间

acos(x)	x 的反余弦角度值，在 0° 到 180° 之间
atan(x)	x 的反正切角度值
atan2(y,x)	y/x 的反正切角度值
trig format=deg|rad	设置三角函数按照角度或弧度模式计算，初值为 deg
equal(x,y)	x 与 y 的平均值
rnd	产生 0 到 1 之间的一个服从均匀分布的随机数
rand	产生 -1 到 1 之间的一个服从均匀分布的随机数
random	与 rnd 相同
random(x)	产生 1 到 x 之间的一个随机整数
random(x,y)	产生 x 到 y 之间的一个随机整数
veclen(x,y)	返回向量 (x,y) 的模 $\sqrt{x^2+y^2}$
min(x1,x2,...,xn)	返回数组的最小值
max(x1,x2,...,xn)	返回数组的最大值
array({x},{y})	用数组 {y} 来索引数组 {x}
sinh(x)	x 的双曲正弦值
cosh(x)	x 的双曲余弦值
tanh(x)	x 的双曲正切值
width(``x'')	返回包含 x 的盒子的宽度
height(``x'')	返回包含 x 的盒子的高度
depth(``x'')	返回包含 x 的盒子的深度

( )，圆括号，定义运算顺序，或者标识函数的自变量，例如，sin(30) 等价于 sin 30（注意空格）; sin 30*10 等价于 sin(30)*10.

{ }，花括号，用来定义数组，数组元素可以是数值也可以是数组，例如 {1, {2,3}, {4,5}, 6}.

[ ]，方括号，索引数组元素，索引顺序是 0,1,2，…

'' ''，双引号，引起文本。

16 用路径算子 plot 绘制函数图像

路径算子 plot 只适合绘制简单的初等函数图。

16.1 两个基本模式：line-to 和 moving-to

```
\path␣.␣.␣.␣--plot<further␣arguments>␣.␣.␣.␣;
\path␣.␣.␣.␣plot<further␣arguments>␣.␣.␣.␣;
```

带两条短连接线的 --plot 是 line-to 模式，即把 plot 之前的路径与 plot 绘制的路径连接起来。不带短线的是 moving-to 模式，从 plot 之前的路径直接跳到 plot 绘制的路径，之间无连接线。

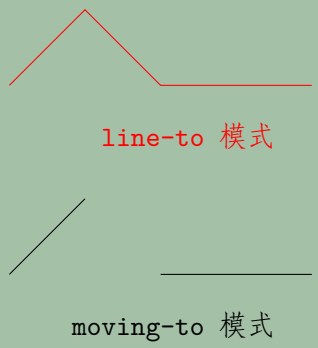
基本句法

```
--plot[<local␣options>]coordinates{<coordinate␣1>
␣␣␣␣␣␣␣<coordinate␣2>␣.␣.␣.␣<coordinate␣n>}
--plot[[<local␣options>]file{<filename>}
--plot[<local␣options>]<coordinate␣expression>
--plot[<local␣options>]function{<gnuplot␣formula>}
```

<coordinate expression> 的意思是，坐标的每个分量都是函数表达式；如果函数表达式中有圆括号，则整个表达式要用花括号括起来。

line-to 和 moving-to 的区别

```
\begin{tikzpicture}
\draw (0,0) -- (1,1) plot coordinates {(2,0) (4,0)}
      node[below left=0.5 ] {moving-to模式};
\draw[color=red,yshift=2.5cm]
      (0,0) -- (1,1) -- plot coordinates {(2,0) (4,0)}
        node[below left=0.5 ]{line-to模式};
\end{tikzpicture}
```



16.2 用坐标点绘制折线图

```
\tikz \draw plot coordinates {(0,0) (1,1) (2,0) (3,1) (2,1) (10:2cm)};
```

16.3 与绘图的有关变量

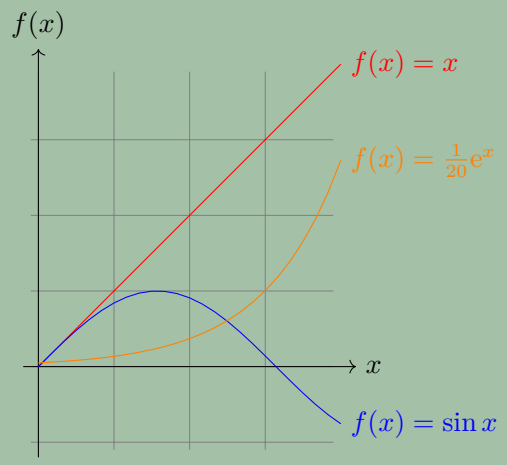
对于函数表达式有以下可用选项

```
variable=<macro> 自定义自变量，必须以宏符号\开头，默认\x
samples=<number> 样本点数目，默认25
domain=<start>:<end> 定义域，默认-5:5
samples at=<sample list> 自变量取值表，用以计算样本点
```

在下面的代码中，符号 \x r 是将角度 \x 转换为弧度

```
\begin{tikzpicture}[domain=0:4]
\draw[very thin,color=gray] (-0.1,-1.1) grid (3.9,3.9);
```

```
\draw[->] (-0.2,0) -- (4.2,0) node[right] {$x$};
\draw[->] (0,-1.2) -- (0,4.2) node[above] {$f(x)$};
\draw[color=red] plot (\x,\x) node[right] {$f(x) =x$};
\draw[color=blue] plot (\x,{sin(\x r)}) node[right] {$f(x) = \sin x$};
\draw[color=orange] plot (\x,{0.05*exp(\x)})
     node[right] {$f(x) = \frac{1}{20} \mathrm e^x$};
\end{tikzpicture}
```



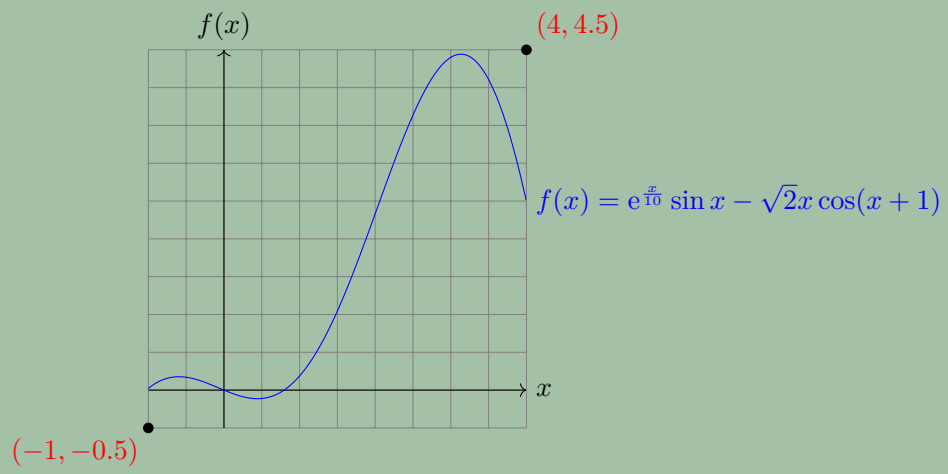
下面的代码画三维参数曲线，选项 smooth 会使得曲线更加平滑。

```
\tikz \draw[domain=0:360,smooth,variable=\t]
        plot ({sin(\t)},\t/360,{cos(\t)});
```

16.4 复合函数举例

下面例子中的运算可能需要调用 math，cal 程序库

```
\begin{tikzpicture}[samples=500,domain=-1:4]
\draw [help lines,step=0.5cm]
     (-1,-0.5)node[below left,red]{$(-1,-0.5)$} grid
     (4,4.5)node[above right,red]{$(4,4.5)$};
\draw[->] (-1,0) -- (4,0) node[right] {$x$};
\draw[->] (0,-0.5) -- (0,4.5) node[above] {$f(x)$};
\draw[color=blue] plot (\x,{exp(\x / 10) * (sin(\x r))
               - (2 ^ (1/2)) * \x * (cos(\x r + 1))})
     node[right] {$f(x) = \mathrm{e}^{\frac x {10}} \sin x
                      - \sqrt{2} x \cos (x+1)$};
\fill (-1,-0.5) circle [radius=2pt]
     (4,4.5) circle [radius=2pt];
\end{tikzpicture}
```

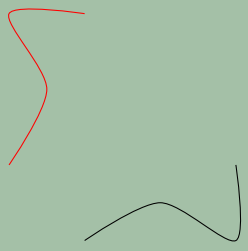


17 坐标轴和图形的变换

17.1 设置坐标轴的单位长度和方向

```
x=<value> 设置x轴的单位长度是多长（注意长度单位）
x={<coordinate>} 设置x轴的单位向量
y=<value> 类似
y={<coordinate>} 类似
z=<value> 类似
z={<coordinate>} 类似
```

```
\begin{tikzpicture}[smooth]
\draw plot coordinates{(1,0) (2,0.5) (3,0) (3,1)};
\draw[x={(0cm,1cm)},y={(1cm,0cm)},color=red]
      plot coordinates{(1,0) (2,0.5) (3,0) (3,1)};
\end{tikzpicture}
```



上面的例子中，选项 x={(0cm,1cm)}，y={(1cm,0cm) 定义了一个新坐标系，然后在这个新坐标系中绘图。

17.2 坐标轴以及图形的变换

下面的变换都是针对图形及其固连坐标系的

```
xshift=<value> 将坐标轴和图一起沿着x轴平移
yshift=<value> 将坐标轴和图一起沿着y轴平移
zshift=<value> 将坐标轴和图一起沿着z轴平移
shift={<coordinate>} 将坐标轴和图一起按指定向量平移
```

```
shift only 只保留复合变换中的平移
scale=<factor> 以原点为位似中心作位似放缩，对文字、字母无影响
scale around={<factor> : <coordinate>} 以指定坐标为位似中心作位似放缩
xscale=<factor> 点的纵标不变，横标放缩factor（有正负）倍
yscale=<factor> 类似
xslant=<factor> 将点(x,y)变成(x+y*factor,y)
yslant=<factor> 类似
rotate=<degree> 以原点为中心旋转
rotate around={<degree>:<coordinate>} 以指定点为中心旋转
rotate around x=<angle> 在三维坐标系中，绕x轴旋转（有向角度）
rotate around y=<angle> 类似
rotate around z=<angle> 类似
cm={<a>,<b>,<c>,<d>,<coordinate>} 这是二维的一般变换
```

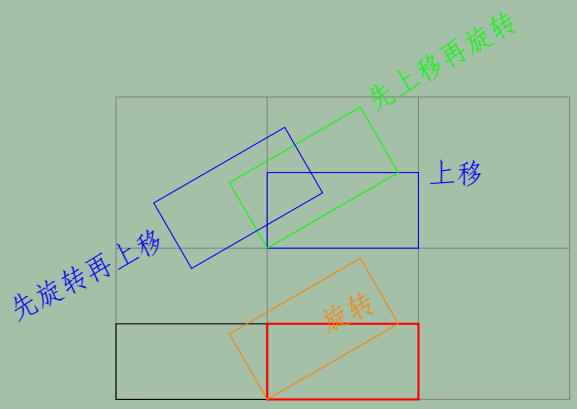
假设对点 $P(p_1,p_2)$ 施加 cm 选项的变换得到点 $P'(p_1',p_2')$，令选项中的 coordinate 是 (m,n)，则变换就是

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} a & b \\ c & d \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} m \\ n \end{pmatrix}$$

17.3 变换的次序

上面的选项是针对图形及其固连坐标系的，故平移和旋转复合时，交换次序可能产生不同结果。

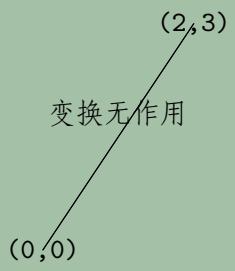
```
\begin{tikzpicture}[scale=2]
\draw[help lines] (0,0) grid (3,2);
\draw (0,0) rectangle (1,0.5);
\begin{scope}[xshift=1cm]
\draw [red, thick] (0,0) rectangle (1,0.5);
\draw[yshift=1cm] [blue] (0,0) rectangle (1,0.5) node[right]{上移};
\draw[rotate=30] [orange] (0,0) rectangle node[sloped,right]{旋转}(1,0.5);
\draw[yshift=1cm,rotate=30] [green] (0,0) rectangle (1,0.5)
       node[rotate=30,right]{先上移再旋转} ;
\draw[rotate=30,yshift=1cm] [blue] (0,0) rectangle
       node[sloped,left=1]{先旋转再上移}(1,0.5) ;
\end{scope}
\end{tikzpicture}
```



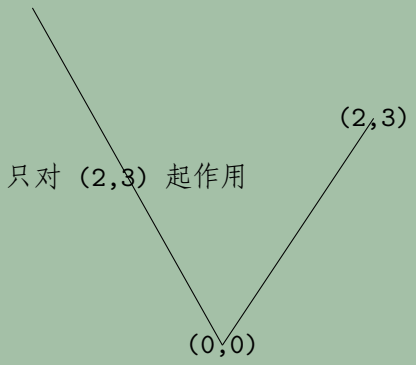
17.4 变换的失效

注意，shift，rotate 选项对坐标名称（即 coordinate 对象的名称）不起作用，而对数值坐标（即“点”）起作用。比较下面语句中的 shift，rotate 选项对路径的作用

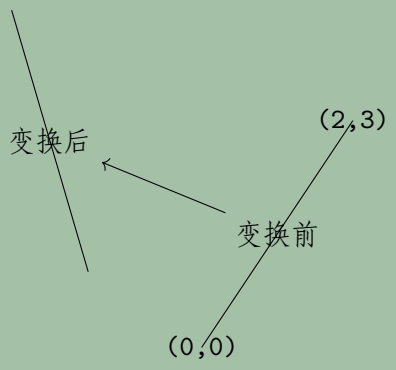
```
\begin{tikzpicture}
\coordinate (a) at (0,0);
\coordinate (b) at (2,3);
\draw [shift={(-1.5,1)},rotate=50]
       (a) -- node[above]{变换无作用} (b);
\draw (0,0) node{(0,0)}--(2,3) node{(2,3)};
\end{tikzpicture}
```




```
\begin{tikzpicture}
\coordinate (a) at (0,0);
\coordinate (b) at (2,3);
\draw [shift={(-1.5,1)},rotate=50]
       (a) -- node{只对(2,3)起作用} (2,3);
\draw (0,0) node{(0,0)}--(2,3) node{(2,3)};
\end{tikzpicture}
```



```
\begin{tikzpicture}
\coordinate (a) at (0,0);
\coordinate (b) at (2,3);
\draw [shift={(-1.5,1)},rotate=50]
      (0,0)--node(hou){变换后}(2,3);
\draw (a) node{(0,0)}
             -- node(qian){变换前}
        (b) node{(2,3)};
\draw [->](qian)--(hou);
\end{tikzpicture}
```



17.5 作一个图形关于某（两点确定的）直线的对称图形

17.5.1 利用 cm 选项

设点 $A(a_1,a_2)$，$B(b_1,b_2)$ 确定直线 AB，点 $P(p_1,p_2)$ 关于直线 AB 的对称点 $P'(p'_1,p'_2)$ 如下

$$\begin{aligned}\mathbf{P}' &= \mathbf{P}-\mathbf{A}-2\left(\mathbf{e},\mathbf{P}-\mathbf{A}\right)\mathbf{e}+\mathbf{A}\\ &= \left(\mathbf{I}-2\left(\mathbf{e},\mathbf{e}^T\right)\right)\left(\mathbf{P}-\mathbf{A}\right)+\mathbf{A}\\ &= \begin{pmatrix}\dfrac{(a_1-b_1)^2-(a_2-b_2)^2}{(a_1-b_1)^2+(a_2-b_2)^2} & \dfrac{2(a_1-b_1)(a_2-b_2)}{(a_1-b_1)^2+(a_2-b_2)^2}\\ \dfrac{2(a_1-b_1)(a_2-b_2)}{(a_1-b_1)^2+(a_2-b_2)^2} & \dfrac{(a_2-b_2)^2-(a_1-b_1)^2}{(a_1-b_1)^2+(a_2-b_2)^2}\end{pmatrix}\begin{pmatrix}p_1-a_1\\ p_2-a_2\end{pmatrix}+\begin{pmatrix}a_1\\ a_2\end{pmatrix}\\ &= \begin{pmatrix}p'_1\\ p'_2\end{pmatrix}\end{aligned}$$

其中 $\mathbf{e}$ 是直线 AB 的单位法向量，变换矩阵只与 A，B 的坐标有关，只要计算出变换矩阵的元素就可以用 cm 选项作轴对称变换。可以调用数学程序库进行计算。

17.5.2 利用 x={<coordinate>}，y={<coordinate>} 和 shift={<coordinate>} 选项

算式

$$\left(\mathbf{I}-2\left(\mathbf{e},\mathbf{e}^T\right)\right)\left(\mathbf{P}-\mathbf{A}\right)$$

表示点 $\mathbf{P}-\mathbf{A}$ 关于参数直线 $t\cdot(\mathbf{A}-\mathbf{B})$ 的对称点 $\overline{\mathbf{P}-\mathbf{A}}$。$\mathbf{P}-\mathbf{A}$ 是对 $\mathbf{P}$ 的平移。

如果作出 xOy 坐标系关于直线 $t\cdot(\mathbf{A}-\mathbf{B})$ 对称的坐标系 $x'Oy'$，那么点 $\overline{\mathbf{P}-\mathbf{A}}$ 在 $x'Oy'$ 系内的坐标与点 $\mathbf{P}-\mathbf{A}$ 在 xOy 坐标系内的坐标相同。

坐标系 $x'Oy'$ 是左手系，可设其单位向量在 xOy 坐标系内的表达式是

$$x'\text{轴单位向量}:\begin{pmatrix}\cos\varphi\\ -\sin\varphi\end{pmatrix}=(-\varphi:1),\quad y'\text{轴单位向量}:\begin{pmatrix}-\sin\varphi\\ -\cos\varphi\end{pmatrix}=(-\varphi-90^\circ:1)$$

设 $\mathbf{P}-\mathbf{A}$ 在 xOy 坐标系内的坐标是 (α,β)，则

$$xOy\text{系}:\mathbf{P}-\mathbf{A}=\alpha\begin{pmatrix}1\\0\end{pmatrix}+\beta\begin{pmatrix}0\\1\end{pmatrix}$$
$$x'Oy'\text{系}:\overline{\mathbf{P}-\mathbf{A}}=\alpha\begin{pmatrix}\cos\varphi\\-\sin\varphi\end{pmatrix}+\beta\begin{pmatrix}-\sin\varphi\\-\cos\varphi\end{pmatrix}$$

先求坐标系 $x'Oy'$ 的单位向量在 xOy 坐标系内的坐标。

参数直线 $t\cdot(\mathbf{A}-\mathbf{B})$ 的方向是 $\mathbf{A}-\mathbf{B}$，如果能求得这个方向与 xOy 系的单位向量之间的夹角，用对称的办法就能容易地知道系 $x'Oy'$ 的单位向量的坐标。再利用 $\mathbf{P}-\mathbf{A}$ 在 xOy 坐标系内的坐标，从而容易确定点 $\overline{\mathbf{P}-\mathbf{A}}$ 的位置。然后再通过平移，即 $+\mathbf{A}$ 得到点 $P(p_1,p_2)$ 关于直线 AB 的对称点 $P'(p_1',p_2')$。

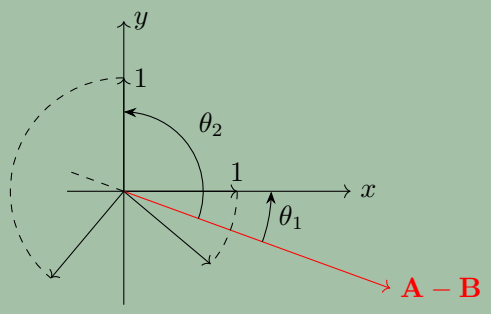
按照这个分析可以作出图形 `F` 关于直线 AB 的对称图形 `F'`。设从方向向量 $\mathbf{A}-\mathbf{B}$ 到单位向量 $(1,0)$ 的夹角是 θ_1，从方向向量 $\mathbf{A}-\mathbf{B}$ 到单位向量 $(0,1)$ 的夹角是 θ_2，则

$$(\mathbf{A}-\mathbf{B})\times(1,0)=|\mathbf{A}-\mathbf{B}|\sin\theta_1=b_2-a_2$$
$$(\mathbf{A}-\mathbf{B})\times(0,1)=|\mathbf{A}-\mathbf{B}|\sin\theta_2=a_1-b_1$$

注意 $0^\circ\leqslant\arccos t\leqslant 180^\circ$，$-90^\circ\leqslant\arcsin t\leqslant 90^\circ$，可以分以下几种情况分析

(i) 如果 $b_2-a_2>0$ 且 $a_1-b_1>0$，则 $\mathbf{A}-\mathbf{B}$ 在第四象限，此时

$$\theta_1=\arcsin\frac{b_2-a_2}{|\mathbf{A}-\mathbf{B}|},\quad \theta_2=\theta_1+90^\circ$$



坐标系 xOy 变为坐标系 $x'Oy'$，则 x 轴的单位向量 $(1,0)$ 变为 x' 轴的单位向量

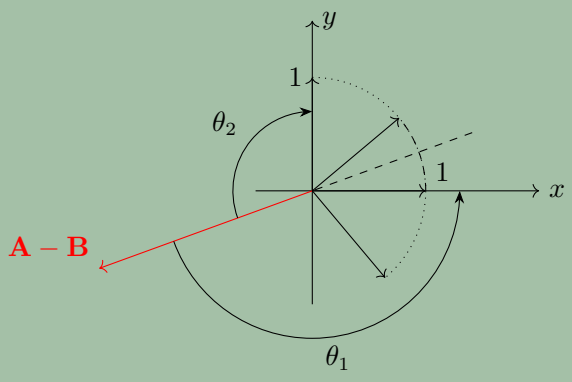
$$(\cos(-2\theta_1),\sin(-2\theta_1))=(\cos 2\theta_1,-\sin 2\theta_1)$$

因为坐标系 $x'Oy'$ 是左手系，故 y 轴的单位向量 $(0,1)$ 变为 y' 轴的单位向量

$$(-\sin 2\theta_1,-\cos 2\theta_1)$$

(ii) 如果 $b_2-a_2>0$ 且 $a_1-b_1<0$，则 $\mathbf{A}-\mathbf{B}$ 在第三象限，此时

$$\theta_1=180^\circ-\arcsin\frac{b_2-a_2}{|\mathbf{A}-\mathbf{B}|},\quad \theta_2=\theta_1-270^\circ$$



x 轴的单位向量 $(1,0)$ 变为 x' 轴的单位向量

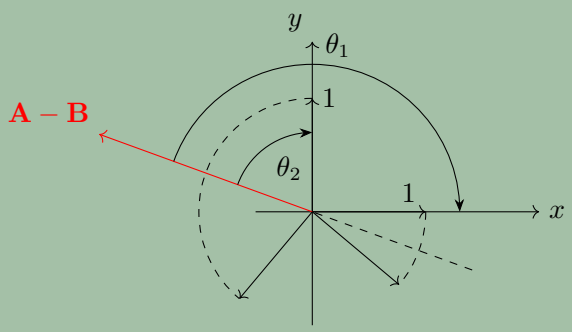
$$(\cos 2(180^\circ-\theta_1),\sin 2(180^\circ-\theta_1))=(\cos 2\theta_1,-\sin 2\theta_1)$$

y 轴的单位向量 $(0,1)$ 变为 y' 轴的单位向量

$$(-\sin 2\theta_1,-\cos 2\theta_1)$$

(iii) 如果 $b_2-a_2<0$ 且 $a_1-b_1<0$，则 $\mathbf{A}-\mathbf{B}$ 在第二象限，此时

$$\theta_2=\arcsin\frac{a_1-b_1}{|\mathbf{A}-\mathbf{B}|},\quad \theta_1=\theta_2-90^\circ=-180^\circ-\arcsin\frac{b_2-a_2}{|\mathbf{A}-\mathbf{B}|}$$



x 轴的单位向量 $(1,0)$ 变为 x' 轴的单位向量

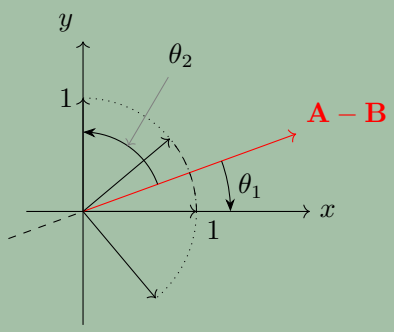
$$(\cos(-2(180^\circ+\theta_1)),\sin(-2(180^\circ+\theta_1)))=(\cos 2\theta_1,-\sin 2\theta_1)$$

y 轴的单位向量 $(0,1)$ 变为 y' 轴的单位向量

$$(-\sin 2\theta_1,-\cos 2\theta_1)$$

(iv) 如果 $b_2-a_2<0$ 且 $a_1-b_1>0$，则 $\mathbf{A}-\mathbf{B}$ 在第一象限，此时

$$\theta_2=\arcsin\frac{a_1-b_1}{|\mathbf{A}-\mathbf{B}|},\quad \theta_1=\theta_2-90^\circ=\arcsin\frac{b_2-a_2}{|\mathbf{A}-\mathbf{B}|}$$



x 轴的单位向量 $(1,0)$ 变为 x' 轴的单位向量

$$(\cos(-2\theta_1),\sin(-2\theta_1)) = (\cos 2\theta_1, -\sin 2\theta_1)$$

y 轴的单位向量 $(0,1)$ 变为 y' 轴的单位向量

$$(-\sin 2\theta_1, -\cos 2\theta_1)$$

(v) 对于 $b_2 - a_2 = 0$ 或 $a_1 - b_1 = 0$ 的情况，显然，x 轴的单位向量 $(1,0)$ 变为 x' 轴的单位向量

$$(\cos(-2\theta_1),\sin(-2\theta_1)) = (\cos 2\theta_1, -\sin 2\theta_1)$$

y 轴的单位向量 $(0,1)$ 变为 y' 轴的单位向量

$$(-\sin 2\theta_1, -\cos 2\theta_1)$$

综上，总有

$$(1,0) \to (\cos 2\theta_1, -\sin 2\theta_1) = (-2\theta_1 : 1)$$
$$(0,1) \to (-\sin 2\theta_1, -\cos 2\theta_1) = (-2\theta_1 - 90^\circ : 1)$$

可以调用数学程序库，计算 θ_1，$\cos 2\theta_1$，$\sin 2\theta_1$。例如，下面模式的代码可以计算 θ_1

```
\tikzmath{
      coordinate \dirvec;
            \dirvec=<(a_1-b_1,a_2-b_2)>;
      real \thetaa;
            \thetaa1=-asin(\dirvecy/veclen(\dirvec));
            \thetaa2=180-\thetaa1;
            \thetaa3=-180-\thetaa1;
            \thetaa4=\thetaa1;
            }
\thetaa1,\thetaa2,\thetaa3,\thetaa4;
```

然后可用下面的代码

```
\draw [shift={<(a_1,a_2)>}]
     [x={(-2<\theta_1>:1)},y={(-2<\theta_1>-90:1)}]
     [shift={($-1*<(a_1,a_2)>$)}]
     < 图形 F 在 xOy 系中的代码，其中要用数值点，不能用 coordinate 名称代替 >;
```

作出图形 F 关于直线 AB 的对称图形 F'。注意，这里必须用三个方括号分别设置选项，而且这三个方括号的次序不能调换。首先是第三个方括号的选项作用于路径，然后是第二个方括号的选项作用，最后是第一个方括号的选项作用。第三个方括号表示对路径上的点作 xOy 系内的平移 $-\mathbf{A}$。第二个方括号表示转换到 $x'Oy'$ 系。第一个方括号表示 xOy 系内的平移 $+\mathbf{A}$。

17.5.3 利用坐标运算

表达式 ($ (A)!(B)!(C) $) 指示点 B 在直线 AC 上的正交射影点（垂足）。

表达式 ($ (A)!x!(B) $) 确定直线 AB 上一个点，该点与点 A 的距离比上 AB 的长度就是数值 x；如果 x 大于 0，该点在射线 AB 上；如果 x 小于 0，该点在射线 BA 上，且点 A 介于该点与点 B 之间。

用这两个算式可作出点 P 关于直线 AB 的对称点。所有对称点构成对称图形。对于简单的图形这个方法比较便捷。

18 预定义的基本几何图形

预定义的几何图形是圆和矩形，调用 shapes.geometric 程序库后有更多图形可用。

18.1 矩形 rectangle

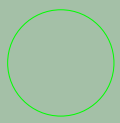
```
<coordinate> rectangle <coordinate> 指定对角线坐标画矩形
选项 rounded corners 把直角改为圆角
选项 rounded corners=10pt 设置圆角半径为10pt
```

```
\tikz \draw (0,0) rectangle (2,2);
\tikz \draw[rounded corners] (0.5,0.5) rectangle (1.5,1.5);
\tikz \draw[rounded corners=10pt] (2,0) -- (2,2) -- (02);
```

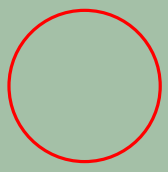
18.2 圆 circle

```
circle 之前是圆心坐标，之后的选项是半径尺寸，半径尺寸可放在方括号内，也可放在圆括号内。
```

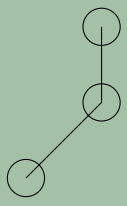
```
\begin{tikzpicture}
\draw[color=green,very thin] (10pt,15pt)
     circle[radius=20pt];
\end{tikzpicture}
```



```
\begin{tikzpicture}
\draw[color=red,very thick] (10pt,15pt)
     circle(1cm);
\end{tikzpicture}
```



```
\begin{tikzpicture}[radius=7pt]
\draw (0,0) circle -- (1,1) circle -- ++(0,1) circle;
\end{tikzpicture}
```



18.3 椭圆 ellipse

若 circle 带有 x radius=，y radius= 选项，就是设置横、纵两个方向的半轴画椭圆，此时可用 ellipse 代替 circle。

```
\begin{tikzpicture}
\fill (1,0) ellipse [x radius=1cm, y radius=5mm, rotate=30];
\end{tikzpicture}
```

算子 ellipse 或 circle 的选项可以采用比较简洁的圆括号形式，例如

```
\tikz \draw (0,0) ellipse (0.3 and 0.5);
```

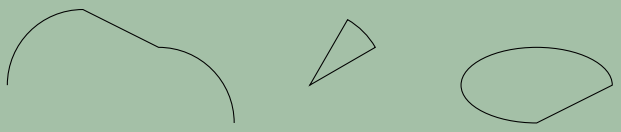
其中圆括号内的各项依次是：横半轴长度 and 纵半轴长度，长度单位默认是 cm，这种简洁形式只能用在圆括号内，而不能用在方括号内，而且圆括号内不能再有其他选项。

18.4 圆弧 arc

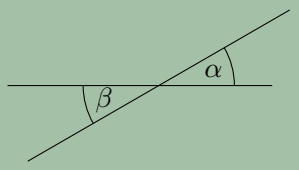
```
<coordinate> arc [<options>]
这里的<coordinate>是起点坐标
选项 start angle=<degrees> 设置起始角度，是相对于圆弧的中心而言的角度
选项 end angle=<degrees> 设置终止角度，是相对于圆弧的中心而言的角度
选项 delta angle=<degrees> 设置终止角度减起始角度之差
选项 radius=尺寸 设置弧的半径尺寸
```

注意比较下面例子中 -- cycle 的作用

```
\begin{tikzpicture}[radius=1cm]
\draw (0,0) arc[start angle=180, end angle=90]-- (2,.5)
      arc[start angle=90, delta angle=-90];
\draw (4,0) -- +(30:1cm)
      arc [start angle=30, delta angle=30] -- cycle;
\draw (8,0) arc [start angle=0, end angle=270,
      x radius=1cm, y radius=5mm] -- cycle;
\end{tikzpicture}
```

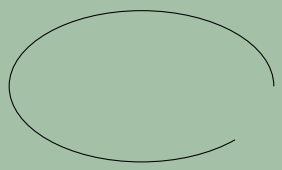


```
\begin{tikzpicture}[radius=1cm,delta angle=30]
\draw (-1,0) -- +(3.5,0);
\draw (1,0) ++(210:2cm) -- +(30:4cm);
\draw (1,0) +(0:1cm) arc [start angle=0];
\draw (1,0) +(180:1cm) arc [start angle=180];
\path (1,0) ++(15:.75cm) node{$\alpha$};
\path (1,0) ++(15:-.75cm) node{$\beta$};
\end{tikzpicture}
```



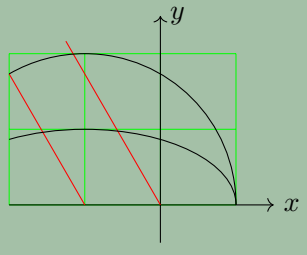
也可以画椭圆弧

```
\tikz \draw (0,0) arc [start angle=0, end angle=315,
          x radius=1.75cm, y radius=1cm];
```



算子 arc 的选项也可以采用比较简洁的圆括号形式，

```
\begin{tikzpicture}
\draw [line width=0.1pt,green](-2,0) grid [step=1] (1,2);
\draw [->](-2,0)--(1.5,0) node[right]{$x$};
\draw [->](0,-0.5)--(0,2.5) node[right]{$y$};
\draw [red](0,0)--(120:2.5);
\draw [red](-1,0)-- ++(120:2);
\draw (1,0) arc (0:120:2);
\draw (1,0) arc (0:120:2 and 1);
\end{tikzpicture}
```



其中圆括号内的各项依次是：起始角度：终止角度：半径，这样画出的是圆弧；或者是：起始角度：终止角度：横半轴长度 and 纵半轴长度，这样画出的是椭圆弧。程序画圆弧时，会根据选项自动确定圆弧的中心。程序画椭圆弧的步骤是，先按比较长的半轴画圆弧，然后固定起始点和圆弧中心不动，再进行伸缩，这样椭圆弧的终止点与圆弧的终止点很可能不一样，不过这两个终止点都在同一竖直线或水平线上。注意这种简洁形式只能用在圆括号内，而不能用在方括号内，而且圆括号内不能再有其他选项。

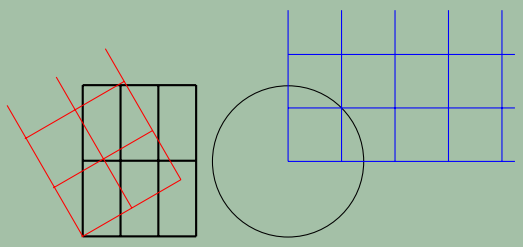
18.5 网格 grid，help lines

```
<coordinate> grid [<options>] <coordinate>;
[<options>]<coordinate> grid <coordinate>;
grid 前后的坐标指示网格所在区域的对角线。
```

```
选项 step=<number or dimension or coordinate> 设置每小格的边长，初始值1cm
选项 step=<coordinate> 指示一个小格的对角线向量
选项 xstep=<dimension or number> 设置横向步长，初始值1cm
选项 ystep=<dimension or number> 设置纵向边长，初始值1cm
```

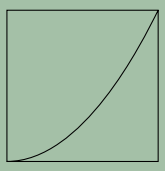
```
help lines 是预定义的网格，初始值 line width=0.2pt,gray
```

```
\begin{tikzpicture}[x=.5cm]
\draw[thick] (0,0) grid [step=1] (3,2);
\draw[red] (0,0) grid [step=.75cm,rotate=30]
                  (3,2);
\end{tikzpicture}
\begin{tikzpicture}
\draw (0,0) circle [radius=1];
\draw[blue] (0,0) grid [step=(45:1)] (3,2);
\end{tikzpicture}
```



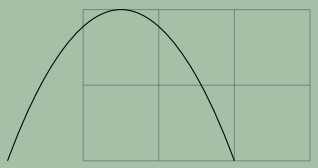
18.6 抛物线算子 parabola

```
\tikz \draw [scale=2](0,0) rectangle (1,1)
          (0,0) parabola (1,1);
```



```
\tikz \draw [x=1pt,y=1pt,scale=4]
          (0,0) parabola bend (4,16) (6,12);
```

```
\begin{tikzpicture}
\draw[help lines] (0,0) grid (3,2);
\draw (-1,0) parabola[parabola height=2cm] +(3,0);
\end{tikzpicture}
```



18.7 正弦，余弦算子 sin，cos

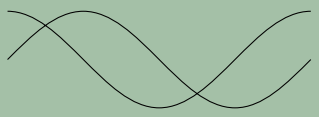
这两个算子只能画出区间 $[0,\pi/2]$ 内的 $\sin(x),\cos(x)$ 图，区间端点为 $\pi/2$ 的整数倍的区间内的图可以用拼接的方法画出。

```
A sine \tikz \draw[x=1ex,y=1ex] (0,0) sin (1.57,1); curve.
```

A sine ⌒ curve.

注意下图中 x 轴的单位

```
\tikz \draw[x=1.57ex,y=1ex,scale=4]
(0,0) sin (1,1) cos (2,0) sin (3,-1) cos (4,0)
(0,1) cos (1,0) sin (2,-1) cos (3,0) sin (4,1);
```



18.8 菱形 diamond

可以将 aspect=ratio 作为环境或者 node 的选项，设置菱形的横宽与竖高之比

```
\tikz [aspect=2]
    \node[shape=diamond ,draw] {Diamond};
其中的shape=可以省略
```

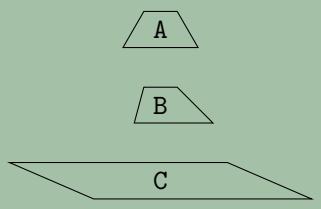
18.9 梯形 trapezium

如果没有其他选项，选项 trapezium 画出等腰梯形，另外有以下选项

```
trapezium left angle=<angle> 左下角度数
trapezium right angle=<angle> 右下角度数
trapezium angle=<angle> 两个底角都是某个度数
trapezium stretches=<boolean> 默认 true
minimum size= 最小尺寸
minimum width= 最小宽度
minimum height= 最小高度
trapezium stretches
```

trapezium stretches 允许在某一个方向伸缩,如果没有这个选项,minimum width 或 minimum height 将改变整体尺寸，而不仅改变横向或竖向尺寸。

```
\begin{tikzpicture}
\tikzstyle{every node}=[trapezium, draw]
       \node at (0,2) {A};
\node[trapezium left angle=75, trapezium right angle=45]
           at (0,1) {B};
\node[trapezium left angle=120, trapezium right angle=60,
           trapezium stretches,minimum width=4cm]
            at (0,0) {C};
\end{tikzpicture}
```



18.10 半圆 semicircle

18.11 正多边形 regular polygon

用选项 regular polygon sides=<integer> 设置多边形的边数

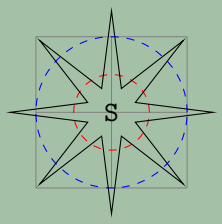
```
\begin{tikzpicture}
\foreach \a in {3,...,7}
      {
       \draw[blue, dashed] (\a*2,0) circle(0.5cm);
       \node[regular polygon, regular polygon sides=\a,
           minimum size=1cm, draw] at (\a*2,0) {};
       }
\end{tikzpicture}
```

18.12 星星 star

star 可以有以下选项

```
star points=<integer> 星星凸出角的个数
star point height=<distance> 点的高度
star point ratio=<number> 角的角度
```

```
\begin{tikzpicture}
\draw [help lines] (0,0) grid (2,2);
\draw [blue, dashed] (1,1) circle(1cm);
\draw [red, dashed] (1,1) circle(.5cm);
\node [star, star points=8,star point ratio=30,
      star point height=1cm,minimum size=2cm, draw]
       at (1,1) {S};
\end{tikzpicture}
```



18.13 等腰三角形 isosceles triangle

等腰三角形有以下选项

```
isosceles triangle apex angle=<angle> 底角度数
isosceles triangle stretches=<boolean> 默认true
```

选项 isosceles triangle stretches 允许在某一个方向伸缩，如果没有这个选项，minimum width 或 minimum height 将改变整体尺寸，而不仅改变横向或竖向尺寸。

18.14 筝形 kite

kite 有以下选项

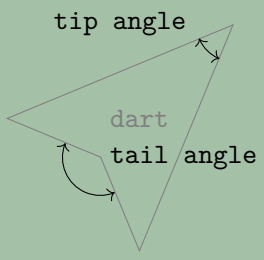
```
kite upper vertex angle=<angle> 顶角度数
kite lower vertex angle=<angle> 底角度数
kite vertex angles=<angle specification> 上下角都是某个度数
```

18.15 飞镖 dart

dart 有两个选项

```
dart tip angle=<angle> 顶角度数
dart tail angle=<angle> 底角度数
```

```
\begin{tikzpicture}
\node[dart, draw, gray, shape border uses incircle,
     shape border rotate=45] (d) {dart};
\draw [<->] (d.tip)++(202.5:.5cm)
      arc(202.5:247.5:.5cm);
\node [left=.5cm] at (d.tip) {tip angle};
\draw [<->] (d.tail center)++(157.5:.5cm)
      arc(157.5:292.5:.5cm);
\node [right] at (d.tail center) {tail angle};
\end{tikzpicture}
```



18.16 扇形 circular sector

扇形有角度选项 circular sector angle=<angle>。

18.17 圆柱 cylinder

cylinder 有以下选项

```
aspect=<value> 底面的圆度，圆度越小就越扁
cylinder end fill=<color> 底面填充颜色
cylinder body fill=<color> 侧面填充颜色
cylinder uses custom fill=<boolean> 能否填充颜色，默认 true
```

```
\begin{tikzpicture}[aspect=0.5]
\node [cylinder, cylinder uses custom fill,
      cylinder end fill=red!50,
      cylinder body fill=red!25] {\huge Cylinder};
\end{tikzpicture}
```

19 象征性图形

首先调用 shapes.symbols 程序库 \usepgflibrary{shapes.symbols}。

在下面的象征性图形中,可以用 aspect,minimum width,minimum height ,fill=yellow,draw=red, line width=2pt 等选项。

19.1 禁止指示图 correct forbidden sign 和 forbidden sign

```
\tikz \node [correct forbidden sign,line width=1ex,
           draw=red,fill=white] {Smoking};

\tikz \node [forbidden sign,line width=1ex,
           draw=red,fill=white] {Smoking};
```

19.2 放大镜 magnifying glass

magnifying glass 有两个相关选项

```
magnifying glass handle angle fill=<degree> 放大镜手柄的角度，默认 -45
magnifying glass handle angle aspect=<factor> 手柄长度与圆半径之比，默认 1.5
```

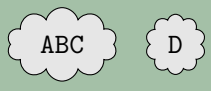
```
\begin{tikzpicture}
\node [magnifying glass,line width=1ex,draw] {huge};
\end{tikzpicture}
```

19.3 云 cloud

cloud 有两个相关选项

```
cloud puffs=<integer> 云瓣个数，初始值 10
cloud puff arc=<angle> 云瓣角度，初始值 135
```

```
\begin{tikzpicture}
\node[cloud, draw, fill=gray!20, aspect=2] {ABC};
\node[cloud, draw, fill=gray!20] at (1.5,0) {D};
\end{tikzpicture}
```



19.4 爆炸图 starburst

starburst 有以下相关选项

```
starburst points=<integer> 突出角的个数，初始值 17
starburst point height=<length> 突出角的高度，初始值 .5cm
```

```
\begin{tikzpicture}
\node[starburst, fill=yellow, draw=red, line width=2pt]
     {\bf BANG!};
\end{tikzpicture}
```

19.5 信号灯 signal

signal 有以下相关选项

```
signal pointer angle=<angle> 终端角度，初始值 90

signal from=<direction> and <opposite direction> 指定内凹端方位，初始值 nowhere，
例如，north，south，east，west，above，below，right，left，nowhere，
nowhere 只有内凹端，没有外凸端

signal to=<direction> and <opposite direction> 指定外凸端方位，初始值 east
```

```
\begin{tikzpicture}
[every node/.style={signal, draw, text=white, signal to=nowhere}]
\node[fill=green!65!black, signal to=east] at (0,1) {To East};
\node[fill=red!65!black, signal from=east] at (0,0) {From East};
\end{tikzpicture}
```

19.6 条带 tape

tape 有以下相关选项

```
tape bend top=<bend style> 指定上边界的弯曲方式，从 in and out，out and in 或 none 之中
三选一，初始值 in and out，将字母 S 逆时针旋转 90 度，就是 in and out 形态

tape bend bottom=<bend style> 指定下边界的弯曲方式，初始值 in and out
```

```
\begin{tikzpicture}
\tikzstyle{every node}=[tape, draw]
\node [tape bend top=out and in, tape bend bottom=out and in]
     {Parallel};
\node at (2,0) [tape bend bottom=out and in] {Why?};
\end{tikzpicture}
```

20 树状图

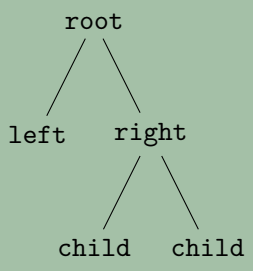
首先调用 trees 程序库 \usetikzlibrary{trees}。

20.1 树的层次构造，在节点间连线

根节点用 \node 语句设置，子节点用 child { <node 语句设置> child { ... }} 这种句型设置。可以根据环境和花括号分组判断节点所在的层次。在设置好最后一个子节点时加分号。程序会按预设在父节点与子节点画线。

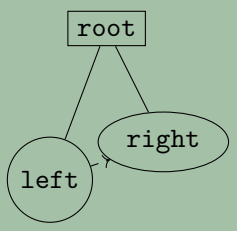
下图中，根之下的层次有 2 个 child，分别命名为 left 和 right；在 right 之下有 2 个 child。

```
\begin{tikzpicture}
\node {root}
 child {node {left}}
 child {node {right}
        child {node {child}}
        child {node {child}}
       };
\end{tikzpicture}
```



可以用连线符号 -- 在子节点之间画线

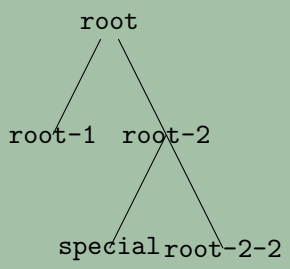
```
\begin{tikzpicture}[sibling distance=15mm]
\node[rectangle,draw] {root}
child {node[circle,draw,yshift=-5mm] (left node) {left}}
child {node[ellipse,draw] (right node) {right}};
\draw[dashed,->] (left node) -- (right node);
\end{tikzpicture}
```



20.2 给节点命名

如前面的例子，可用 node 语句给节点命名。如果不给节点命名，则系统会自动给节点命名。假设父节点的名称是 parent，则它的子节点分别命名为 parent-1，parent-2 等等；parent-1 的子节点分别命名为 parent-1-1，parent-1-2 等等。

```
\begin{tikzpicture}[sibling distance=15mm]
\node (root) {root}
     child
     child {
            child {coordinate (special)}
            child
           };
\node at (root-1) {root-1};
\node at (root-2) {root-2};
\node at (special) {special};
\node at (root-2-2) {root-2-2};
\end{tikzpicture}
```



20.3 节点选项的有效范围

选项的有效范围按如下原则设置

- 对整个树的选项设置应在根之前，放在方括号里。
- 对根节点的选项设置在根的 node 语句中设置。
- 对根下所有子节点的选项设置放在根与第 1 个子节点之间，放在方括号里。
- 在某个子节点后的方括号里（花括号之前）的选项，对该子节点以及该子节点产生的分支都有效。
- 对单个子节点的选项设置应在该子节点后的花括号里用 node 语句设置。
- 特定层次、分支或节点的属性，都可以在本地设置，并且本地设置优于前面的设置。

```
\begin{tikzpicture}
\scoped % 开启子环境画树图
 [...] % 方括号内的选项用于整个树
\node[...] {root} % 方括号内的选项只针对根节点
    [...] % 方括号内的选项针对所有子节点
    child[...] % 方括号内的选项针对该子节点以及该子节点之下的分支
        {
          node[...] {} % 方括号内的选项只针对当前的子节点
...
         }
    child[...] % 方括号内的选项针对该子节点以及该子节点之下的分支
;
\end{tikzpicture}
```

20.4 树层样式设置

可以为环境的所有树层设置样式选项，格式为

```
level/.style={ ... }
```

```
\begin{tikzpicture}[level/.style={sibling distance=20mm/#1}]
\node {root}
    child { child child }
    child { child child child };
\end{tikzpicture}
```

也可以对特定的树层定义样式，下面分别定义第 1 层和第 2 层的样式

```
\begin{tikzpicture}
[level 1/.style={sibling distance=20mm},
level 2/.style={sibling distance=5mm}]
\node {root}
    child { child child }
    child { child child child };
\end{tikzpicture}
```

20.5 树层间距以及一层之内的节点间距

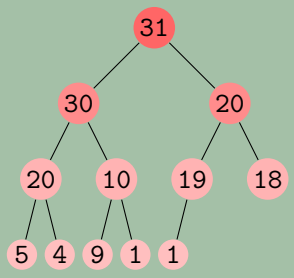
```
level distance=<distance>
按有效范围设置层间距 。如果本选项针对单个节点，则只对从该节点出发的向下的间距有效。
```

```
sibling distance=<distance>
按有效范围，设置层内各节点之间的间距。
```

```
\begin{tikzpicture}
  [level distance=10mm,
    every node/.style={fill=red!60,circle,inner sep=1pt},
    level 1/.style={sibling distance=20mm,nodes={fill=red!45}},
    level 2/.style={sibling distance=10mm,nodes={fill=red!30}},
    level 3/.style={sibling distance=5mm,nodes={fill=red!25}}]

\node {31}
    child {node {30}
          child {node {20}
                child {node {5}}
                child {node {4}}
               }
```

```
          child {node {10}
                child {node {9}}
                child {node {1}}
               }
          }
    child {node {20}
           child {node {19}
                  child {node {1}}
                  child[missing]
                 }
           child {node {18}}
          };
\end{tikzpicture}
```



20.6 节点生长方向选项

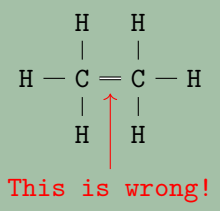
```
grow=<direction>
```

<direction> 可以是某个角度（方向角），也可以是 down, up, left, right, north, south, east, west, north east, north west, south east, south west.

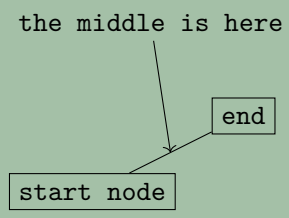
```
grow'=<direction>
```

撇号' 等效于选项 swap，使得与 grow 反向。

```
\begin{tikzpicture}[level distance=2em]
\node {C}
     child[grow=up] {node {H}}
     child[grow=left] {node {H}}
     child[grow=down] {node {H}}
          child[grow=right] {node {C}
              child[grow=up] {node {H}}
              child[grow=right] {node {H}}
              child[grow=down] {node {H}}
          edge from parent[double]
              coordinate (wrong)
                          };
\draw[<-,red] ([yshift=-2mm]wrong) -- +(0,-1)
    node[below]{This is wrong!};
\end{tikzpicture}
```



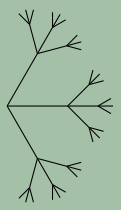

```
\begin{tikzpicture}
\node[rectangle,draw] (a) at (0,0) {start node};
\node[rectangle,draw] (b) at (2,1) {end};
\draw (a) -- (b)
     node[coordinate,midway] {} % 属性是坐标，在路径中点
     child[grow=100,<-] {node[above] {the middle is here}};
\end{tikzpicture}
```



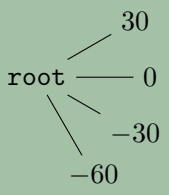
20.7 旋转排布子节点

grow cyclic 使得子节点围绕父节点成环形排布
sibling angle=<angle> 设置一层之内，相邻两条（父-子）连线之间的夹角
clockwise from=<angle> 从角度 <angle> 开始，按顺时针方向排布子节点
counterclockwise from=<angle> 从角度 <angle> 开始，按逆时针方向排布子节点

```
\begin{tikzpicture}
[grow cyclic,
level 1/.style={level distance=8mm,sibling angle=60},
level 2/.style={level distance=4mm,sibling angle=45},
level 3/.style={level distance=2mm,sibling angle=30}]
\coordinate [rotate=-90] % going down
child foreach \x in {1,2,3}
{child foreach \x in {1,2,3}
{child foreach \x in {1,2,3}}};
\end{tikzpicture}
```



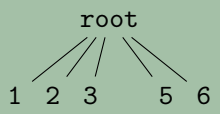
```
\begin{tikzpicture}
\node {root}
[clockwise from=30,sibling angle=30]
       child {node {$30$}}
       child {node {$0$}}
       child {node {$-30$}}
       child {node {$-60$}};
\end{tikzpicture}
```



20.8 隐形占位 missing

```
missing=<true or false> 默认 true
节点选项 missing 或 missing=true 隐藏该节点，但占据显示位置
```

```
\begin{tikzpicture}[level distance=10mm,sibling distance=5mm]
\node {root} [grow=down]
     child { node {1} }
     child { node {2} }
     child { node {3} }
     child[missing] { node {4} }
     child { node {5} }
     child { node {6} };
\end{tikzpicture}
```



20.9 父子节点之间的连线设置

路径生成算子 edge 可用来设置父子节点之间的连线样式。

```
edge from parent 初始值 draw
```

默认画出父子节点之间的连线

edge from parent [<options>]
选项可以是线型（如 dashed），draw=none，颜色，箭头，标签等等

edge from parent path={<path>}
方括号里可以定义复杂的线型

child anchor=<anchor> 初始值 border
连线所指向的子节点的方位

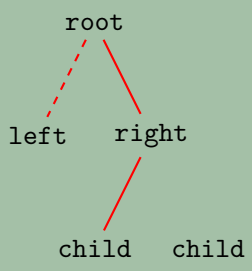
parent anchor=<anchor> 初始值 border
连线所指向的父节点的方位

环境选项

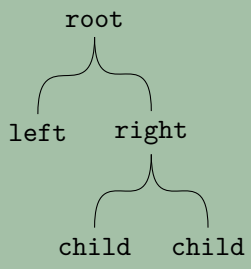
edge from parent/.style={ }

设置环境内的连线样式。

```
\begin{tikzpicture}
     [edge from parent/.style={draw,red,thick}]
\node {root}
     child {node {left} edge from parent[dashed]}
     child {node {right}
           child {node {child}}
           child {node {child} edge from parent[draw=none]}
          };
\end{tikzpicture}
```



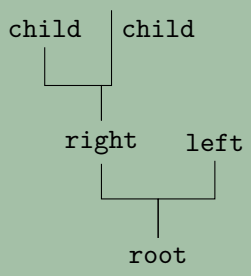

```
\begin{tikzpicture}[level distance=15mm, sibling distance=15mm,
edge from parent path=
{(\tikzparentnode.south) .. controls +(0,-1) and +(0,1)
.. (\tikzchildnode.north)}]
\node {root}
      child {node {left}}
      child {node {right}
            child {node {child}}
            child {node {child}}
            };
\end{tikzpicture}
```



20.10 几种预定义父子节点连线线型

```
edge from parent fork down
edge from parent fork right
edge from parent fork left
edge from parent fork up
```

```
\begin{tikzpicture}
\node {root}
[edge from parent fork down]
   child {node {left}}
   child {node {right}
      child[child anchor=north east] {node {child}}
      child {node {child}}
         };
\end{tikzpicture}
```



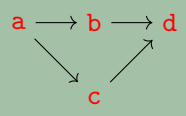
21 图

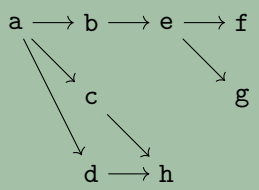
首先调用程序库 \usetikzlibrary{graphs}

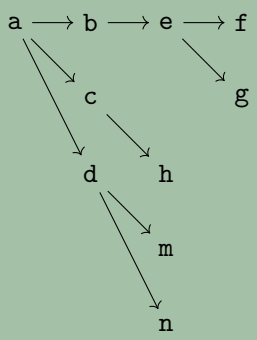
图是由结点和连接两个结点的边组成的集合。\graph 命令可以构造 node 结点，将结点按某种规则排布，并在结点之间画边构成图。它可以省去频繁利用\node 命令构造结点的麻烦。

观察下面例子中花括号分组对两组结点之间的连线规则的影响

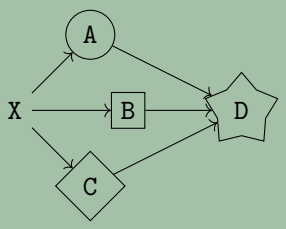
```
\tikz \graph [nodes=red] { a -> {b, c} -> d };
\tikz \graph { a -> {b,c,d} -> {e -> {f,g}, h} };
\tikz \graph { a -> {b,c,d} -> {e -> {f,g}, h,g} };
\tikz \graph { a -> {b,c,d} -> {e -> {f,g}, h, m, n} };
```







```
\tikz [new set=my nodes] {
\node [set=my nodes, circle, draw] at (1,1) {A};
\node [set=my nodes, rectangle, draw] at (1.5,0) {B};
\node [set=my nodes, diamond, draw] at (1,-1) {C};
\node (d) [star, draw] at (3,0) {D};
\graph { X -> (my nodes) -> (d) };
}
```



21.1 两种节点排布规则及其转换关系

默认的结点排布规则是 Cartesian placement.

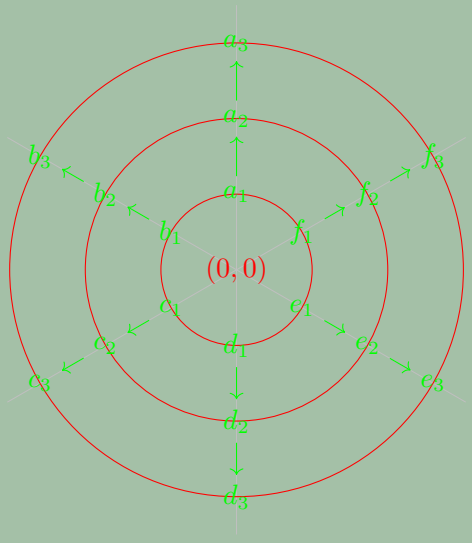
将结点作为（平面第四象限及其边界的）网格上的整数点，每个点都有逻辑宽度（logical width，即笛卡尔坐标的横坐标）和逻辑深度（logical depth，笛卡尔坐标的纵坐标的相反数）两个逻辑坐标。最左上角点（就是坐标系的原点）的 logical width 和 logical depth 都是 0，向右一格 logical width 加 1，向下一格 logical depth 加 1.

circular placement 排布规则

以原点为圆心,以正整数 n 为半径作圆 C_n;设 α 为参数,m 是非负整数;以原点为始点,$90°+m\alpha°$ 为方向作射线 $(90+m\alpha:t)(t\geqslant 0)$，与圆 C_n 相交于点 C_{nm}，则点 C_{nm} 的 logical width 是 $n-1$，logical depth 是 m。这相当于以射线 $(90:t)$ 为初始边构造一种极坐标系。如果没有其他相关选项（如后文的 group polar shift=(angle:radius)），默认 $\alpha=60$。

两种排布规则相互转换时,其中点的位置的变换关系是:在 Cartesian placement 排布规则下 logical 坐标为 (i,j) 的格点对应 circular placement 排布规则下的点 $C_{(i+1)j}$，逻辑坐标仍然是 (i,j)。

```
\begin{tikzpicture}[ray/.style={gray!50}]
\draw [ray](-3.5,0) -- (3.5,0) (0,-3.5)--(0,3.5);
\draw [ray,rotate=60](-3.5,0) -- (3.5,0) (0,-3.5)--(0,3.5);
\draw [ray,rotate=-60](-3.5,0) -- (3.5,0) (0,-3.5)--(0,3.5);
\draw [red](0,0)node[red]{$(0,0)$} circle [radius=1];
\draw [red](0,0) circle [radius=2];
\draw [red](0,0) circle [radius=3];
\graph [math nodes,circular placement,nodes=green,edge=green] {
a_1 -> a_2 -> a_3 -> a_4 -> a_5;
b_1 -> b_2 -> b_3 -> b_4 -> b_5;
c_1 -> c_2 -> c_3 -> c_4 -> c_5;
d_1 -> d_2 -> d_3 -> d_4 -> d_5;
e_1 -> e_2 -> e_3 -> e_4 -> e_5;
f_1 -> f_2 -> f_3 -> f_4 -> f_5;
};
\end{tikzpicture}
```



21.2 graph 的选项

21.2.1 修饰节点和边的选项

```
nodes=<options>
edges=<options> 等于 edge=<options>
edge node=<node specification>
edge label=<text>
edge label'=<text>
edge quotes=<options>
edge quotes center 等于 edge quotes to anchor=center
edge quotes mid 等于 edge quotes to anchor=mid
-> [<options>]
-- [<options>]
<- [<options>]
<-> [<options>]
-!- [<options>] 不画线
```

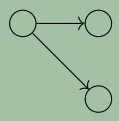
```
as=<text> 将节点显示为文字符号
```

```
\tikz \graph { a [as=$x$] -- b [as=$y_5$] -> c [red, as={a--b}] };
```

x — y_5 → a--b

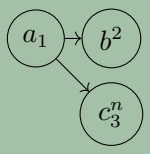
```
empty nodes 结点只占位，不显示
```

```
\tikz \graph [empty nodes, nodes={circle, draw}] { a -> {b, c} };
```



```
math nodes 将符号按数学模式显示
```

```
\tikz \graph [math nodes, nodes={circle, draw}] { a_1 -> {b^2, c_3^n} };
```



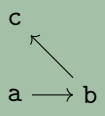
multi 允许两个节点之间画多条线
simple 使两个节点之间只能画一条线

```
\tikz \graph [multi] { % "multi" is not really necessary here
a ->[bend left, red] b;
a ->[bend right, blue] b;
};
```

21.2.2 结点排布形态、样式选项

no placement 启动手工指定结点位置模式。

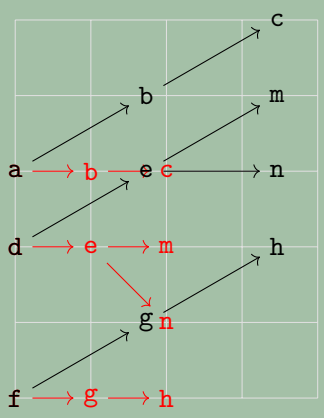
```
\tikz \graph [no placement]
{
a[at={(0:0)}] -> b[at={(1,0)}] -> c[yshift=1cm];
};
```




```
chain shift=<(a:r)>
```

这里 <(a:r)> 必须是极坐标。在这个选项之下，用 logical width 和 logical depth 定义结点坐标，假设原来的结点 $P(p_w, p_d)$ 平移到 P'，再设点 $Q(0, p_d)$（即 Q 与 P 处于同一水平线上，且在最左边），则 $P'Q = r \cdot PQ$，有向角$\angle PQP' = a^\circ$。

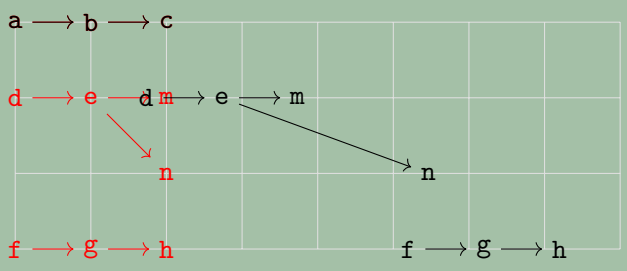
```
\begin{tikzpicture}
\draw [gray!25](0,-3)grid(4,2);
\graph [nodes=red,edge=red]{
a -> b -> c;
d -> e -> {m,n};
f -> g -> h;
};
\graph [chain shift=(30:2)] {
a -> b -> c;
d -> e -> {m,n};
f -> g -> h;
};
\end{tikzpicture}
```




```
group shift=<(a:r)>
```

这里 <(a:r)> 必须是极坐标。在这个选项之下，用 logical width 和 logical depth 定义结点坐标，假设原来的结点 $P(p_w, p_d)$ 平移到 P'，再设点 $Q(p_w, 0)$（即 Q 与 P 处于同一竖直线上，且在最上边），则 $P'Q = r \cdot PQ$，有向角$\angle PP'Q = a°$。

```
\begin{tikzpicture}
\draw [gray!25](0,-3)grid(8,0);
\graph [nodes=red,edge=red]{
a -> b -> c;
d -> e -> {m,n};
f -> g -> h;
};
\graph [ group shift=(-30:2)] {
a -> b -> c;
d -> e -> {m,n};
f -> g -> h;
};
\end{tikzpicture}
```

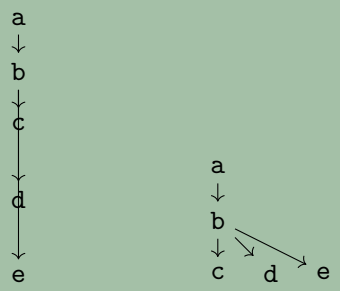



```
grow up=<distance> 默认 1，向上伸展并设置相邻节点的中心间距
grow down=<distance> 默认 1
grow left=<distance> 默认 1
grow right=<distance> 默认 1
branch up=<distance> 默认 1，分支向上并设置相邻节点的中心间距
branch down=<distance> 默认 1
branch left=<distance> 默认 1
branch right=<distance> 默认 1
```

```
grow up sep=<distance> 默认 1em，向上伸展并设置相邻节点的边界间距
grow down sep=<distance> 默认 1em
grow left sep=<distance> 默认 1em
grow right sep=<distance> 默认 1em
branch up sep=<distance> 默认 1em
branch down sep=<distance> 默认 1em
branch left sep=<distance> 默认 1em
branch righ sept=<distance> 默认 1em
circular placement 圆形排布，排布规则，见前文
chain polar shift=(angle:radius) 初始值 (0:1)，在circular placement下有效
group polar shift=(angle:radius) 初始值(45:0)，在circular placement下有效
clockwise=number 将 number 个结点按顺时针排布，在circular placement下有效
counterclockwise=number 将 number 个结点按逆时针排布，在circular placement下有效
radius=<dimension> 初始值 1cm，圆形排布的半径，在circular placement下有效
level 同一宽度的结点作为一层，设置层的样式
level 1/.style={ } 设置第 1 层样式
level 2/.style={ } 设置第 2 层样式
level 3/.style={ } 设置第 3 层样式
```

注意下面例子中 grow 和 branch 命令的配合

```
\tikz \graph [grow down=7mm] { a -> b -> {c, d, e}};
\tikz \graph [branch right=7mm, grow down=7mm] { a -> b -> {c, d, e}};
```

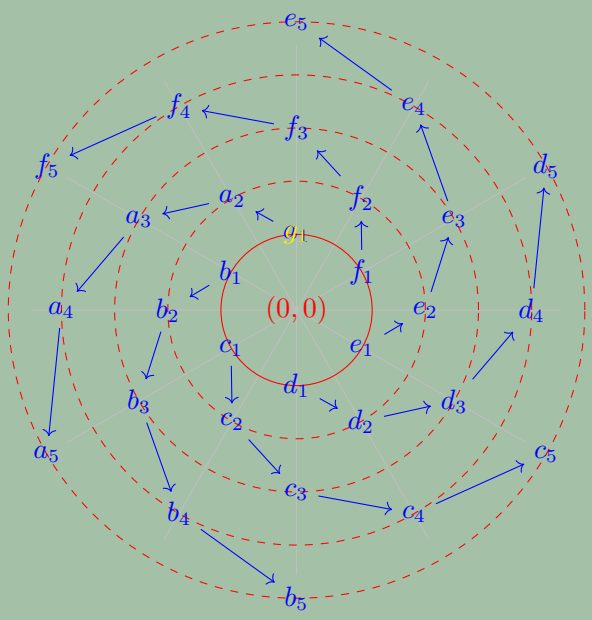



```
chain polar shift=(a:r)
```

这里 <(a:r)> 必须是极坐标，本选项在 circular placement 下才有效。假设原来的结点 $P(i,j)$ 平移到 P'，原点是 O（故 $OP=i+1$），则有向角 $\angle POP'=i\cdot a°$，$P'O=(r\cdot i)\mathtt{pt}+1\mathtt{cm}$。观察下图

```
\begin{tikzpicture}[ray/.style={gray!50}]
\draw [ray](-3.5,0) -- (3.5,0) (0,-3.5)--(0,3.5);
\draw [ray,rotate=60](-3.5,0) -- (3.5,0) (0,-3.5)--(0,3.5);
\draw [ray,rotate=-60](-3.5,0) -- (3.5,0) (0,-3.5)--(0,3.5);
\draw [red](0,0)node[red]{$(0,0)$} circle [radius=1];
\draw [dashed,red](0,0) circle [radius=1cm+20pt];
\draw [dashed,red](0,0) circle [radius=1cm+40pt];
```

```
\draw [dashed,red](0,0) circle [radius=1cm+60pt];
\draw [dashed,red](0,0) circle [radius=1cm+80pt];
\graph [math nodes,circular placement,chain polar shift=(30:20),nodes=blue,edge=blue] {
a_1 -> a_2 -> a_3 -> a_4 -> a_5;
b_1 -> b_2 -> b_3 -> b_4 -> b_5;
c_1 -> c_2 -> c_3 -> c_4 -> c_5;
d_1 -> d_2 -> d_3 -> d_4 -> d_5;
e_1 -> e_2 -> e_3 -> e_4 -> e_5;
f_1 -> f_2 -> f_3 -> f_4 -> f_5;
g_1[yellow];
};
\end{tikzpicture}
```

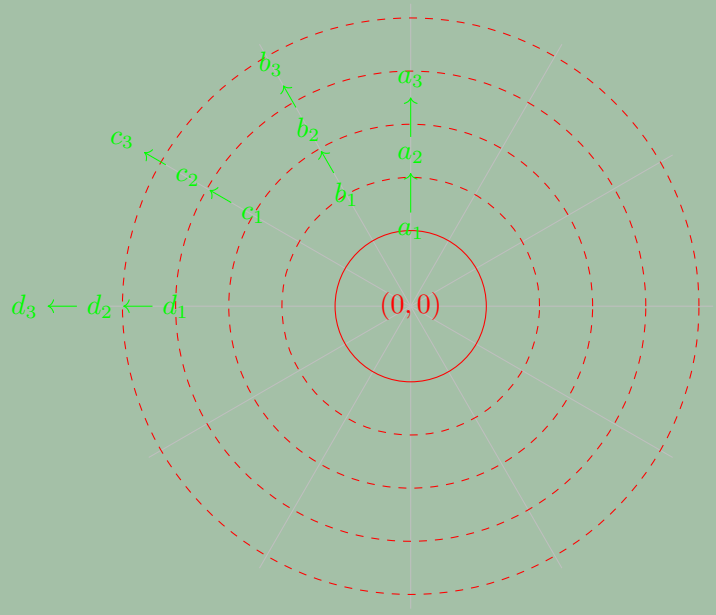



```
group polar shift=(a:r)
```

这里 <(a:r)> 必须是极坐标，本选项在 circular placement 下才有效。设置 circular placement 排布规则中的 α = a。假设原来的结点 $P(i,j)$ 平移到 P'，原点是 O（故 $OP = i+1$），则有向角 $\angle POP' = i \cdot a°$，$P'O = (r \cdot j)\mathtt{pt} + 1\mathtt{cm}$。观察下图

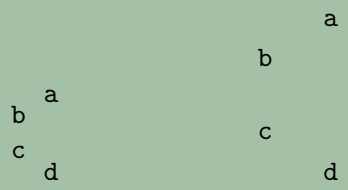
```
\begin{tikzpicture}[ray/.style={gray!50}]
\draw [ray](-4,0) -- (4,0) (0,-4)--(0,4);
\draw [ray,rotate=60](-4,0) -- (4,0) (0,-4)--(0,4);
\draw [ray,rotate=-60](-4,0) -- (4,0) (0,-4)--(0,4);
\draw [red](0,0)node[red]{$(0,0)$} circle [radius=1];
\draw [dashed,red](0,0) circle [radius=1cm+20pt];
\draw [dashed,red](0,0) circle [radius=1cm+40pt];
\draw [dashed,red](0,0) circle [radius=1cm+60pt];
```

```
\draw [dashed,red](0,0) circle [radius=1cm+80pt];
\graph [math nodes,circular placement,nodes=green,edge=green,
        group polar shift=(30:20)] {
a_1 -> a_2 -> a_3;
b_1 -> b_2 -> b_3;
c_1 -> c_2 -> c_3;
d_1 -> d_2 -> d_3;
};
\end{tikzpicture}
```



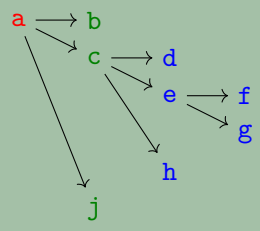
radius 示例

```
\tikz \graph [circular placement, radius=5mm] { a, b, c, d };
\tikz \graph [circular placement, radius=1cm] { a, b, c, d };
```



level 示例

```
\tikz \graph [
branch down=5mm,
level 1/.style={nodes=red},
level 2/.style={nodes=green!50!black},
level 3/.style={nodes=blue}]
{
 a -> {
       b,
       c -> {
            d,
            e -> {f,g},
            h
           },
       j
      }
};
```



22 输出单独的图形文件

22.1 方法一：用 pgfgraphicnamed 命令

按照如下步骤

1. 在导言区设置 \pgfrealjobname{<name>}，其中 <name> 是正在编辑的包含图形的.tex 文件的名称（文档排版保存时必须用这个名称）。例如，编辑的文件名称是 survey.tex，那么可以写下 \pgfrealjobname{survey}。

2. 将需要外部化的图形代码放在宏命令 \beginpgfgraphicnamed{<file name prefix>} 和 \endpgfgraphicnamed 之间。在这两个命令之间只能有一个 *{tikzpicture }* 或 *{pgfpicture}* 环境，其内容将被装入一个 TEX 盒子之中并输出，输出的图形尺寸就是盒子尺寸。

   \beginpgfgraphicnamed{<file name prefix>}，定义待输出的图形文件名，名称的主要部分必须必须与图形所在的.tex 文件的名称一致，不能有模糊符号。例如，如果在导言区设置了 \pgfrealjobname{survey}，
   那么可以给图形命名
   \beginpgfgraphicnamed{survey-f1} 或者
   \beginpgfgraphicnamed{survey-f2}。

3. 排版输出.tex 文件后，在命令行进入.tex 文件所在文件夹，运行

```
bash> latex --jobname=survey-f1 survey.tex
bash> dvips survey-f1
```

得到名称为 survey-f1.ps 的 ps 格式的图形。如果运行

```
bash> pdflatex --jobname=survey-f1 survey.tex
```

得到名称为 survey-f1.pdf 的 pdf 格式的图形。

举例，先执行以下内容

```
\documentclass{article}
\usepackage{graphics}
\usepackage{tikz}
\pgfrealjobname{survey} % 定义文档名称是 survey.tex
\begin{document}
下图是个圆:
\beginpgfgraphicnamed{survey-f1} % 定义第一个图形的名称，注意要与文档名称一致
\begin{tikzpicture}
\fill (0,0) circle (10pt);
\end{tikzpicture}
\endpgfgraphicnamed
作为比较，这是个矩形:
\beginpgfgraphicnamed{survey-f2}
\begin{tikzpicture}
\fill (0,0) rectangle (10pt,10pt);
\end{tikzpicture}
\endpgfgraphicnamed
\end{document}
```

执行后得到文档 survey.tex，然后在命令行进入 survey.tex 文件所在文件夹，如下运行 pdflatex 两次: 运行

```
bash> pdflatex --jobname=survey-f1 survey.tex
```

得到图形 survey-f1; 运行

```
bash> pdflatex --jobname=survey-f2 survey.tex
```

得到图形 survey-f2。

用这种方式得到的 PDF 图形有白边，不够美观。下一种方法就没有这个问题。

22.2 方法二：调用 external 程序库

调用程序库 \usetikzlibrary{external}。这个程序库以自动或半自动的方式输出 TikZ 图形（包括图形的标签等），不必改变文档就可以得到 PDF 格式的图形。

这个程序库以 \begin{tikzpicture} 开始，以 \end{tikzpicture} 结束，不会主动分析这些环境命令的套嵌结构，所以一个环境最好只绘制一个图。

在导言区设置

```
\usepackage{tikz}
\usetikzlibrary{external}
\tikzexternalize
```

\tikzexternalize 用于激活输出图形的功能。

示例

```
\documentclass{article}
% 主文件，名称为 main.tex
\usepackage{tikz}
\usetikzlibrary{external}
\tikzexternalize % 激活
\begin{document}
\begin{tikzpicture}
\node {root}
      child {node {left}}
      child {node {right}
            child {node {child}}
            child {node {child}}
            };
\end{tikzpicture}
A simple image is \tikz \fill (0,0) circle(5pt);.
\end{document}
```

将这一段代码排版，保存名称为 main，会生成 main.tex 文件（控制台可能有问题提示，但只要生成了.tex 文件，就可以忽略控制台的问题提示。）。然后在命令行进入 main.tex 文件所在文件夹，运行

```
pdflatex -shell-escape main
```

就得到名称分别是 main-figure0 和 main-figure1 的 PDF 格式的图形文件。图形文件的名称格式默认为 <real file name>-figure_<number>，编号 <number> 从 0 开始。

22.3 设置文件名、图形名的选项

```
\tikzexternalize[<optional arguments>]{<main job name>}
激活外部化功能，放在导言区。<main job name> 是编辑的 .tex 文件的名称。
```

```
\tikzexternalrealjob{文件名}
调用程序库后，（在导言区）可用该选项定义主文件名。
```

```
\pgfactualjobname{ }
使用 \tikzexternalize 后，（在导言区）可用该选项定义输出的图形名称的主要部分。
```

prefix={<file name prefix>}
作为 \tikzexternalize 的选项，初始值 empty，设置图形文件名的前缀。

\tikzsetexternalprefix{<file name prefix>}
用在 {tikzpicture} 环境之前或绘图命令 \tikz 之前，定义它后面所有输出的图形文件名称的前缀，例如
\tikzsetexternalprefix{figures/}
可以给输出的图形文件名称加前缀 figures/。

\tikzsetnextfilename{<file name>}
设置下一图形的名称，前缀会自动加到这个名称之前。

例子

```
\documentclass{article}
% 主文件，名称为 main.tex
\usepackage{tikz}
\usetikzlibrary{external}
\tikzexternalize[prefix=figures/] % 激活并给图形文件名加前缀figures/
\begin{document}
\tikzsetnextfilename{trees} % 设置下一图形的名称为 trees
\begin{tikzpicture} % 该环境生成的图形名称将是 figures/trees.pdf
\node {root}
      child {node {left}}
      child {node {right}
            child {node {child}}
            child {node {child}}
            };
\end{tikzpicture}
\tikzsetnextfilename{simple} % 设置下一图形的名称为 simple
A simple image is \tikz \fill (0,0) circle(5pt);. % 该图形名称将是 figures/simple.pdf
\begin{tikzpicture} % 该环境生成的图形名称将是 figures/main-figure0.pdf
\draw[help lines] (0,0) grid (5,5);
\end{tikzpicture}
\end{document}
```

figure name={<name>}
作为 \tikzexternalize 的选项，设置该命令之后的所有图形文件的名称。

\tikzsetfigurename{<name>}
设置该命令之后的所有图形文件的名称，也可以用命令
\tikzset{external/figure name={<name>}}

对不同的名称会运用不同的计数器来给图形文件编号，所有编号从 0 开始。设置的名称的有效范围受到花括号分组的限制。

例子

```
\documentclass{article}
% 主文件，名称为 main.tex
\usepackage{tikz}
\usetikzlibrary{external}
\tikzexternalize % 激活
\begin{document}
\begin{tikzpicture} % 该环境生成的图形名称将是 main-figure0.pdf
\node {root}
      child {node {left}}
      child {node {right}
            child {node {child}}
            child {node {child}}
            };
\end{tikzpicture}
{
\tikzsetfigurename{subset_}
A simple image is \tikz \fill (0,0) circle(5pt);. % 该图形名称将是 subset_0.pdf'
\begin{tikzpicture} % 该环境生成的图形名称将是 subset_1.pdf
\draw[help lines] (0,0) grid (5,5);
\end{tikzpicture}
} % 名称 subset_ 至此失效，后面将用原来的名称并继续编号
\begin{tikzpicture} % 该环境生成的图形名称将是 main-figure1.pdf
\draw (0,0) -- (5,5);
\end{tikzpicture}
\end{document}
```