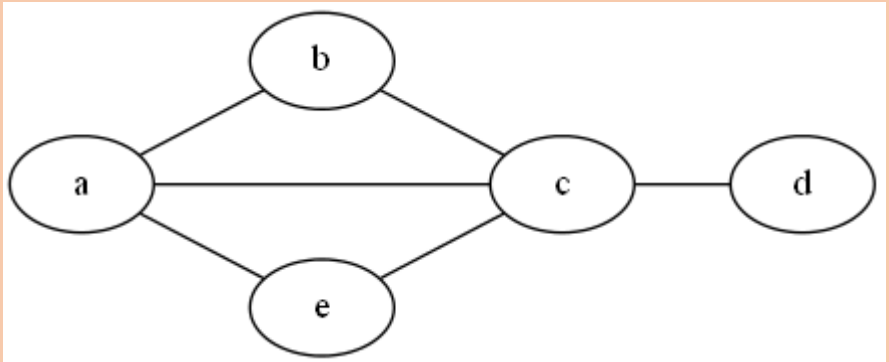
1 graphViz

参考：http://graphs.grevian.org/example

1.1 无向图

```
graph {
    rankdir=LR
    a -- b;
    a -- c;
    a -- e;

    b -- c;
    e -- c;

    c -- d;
}
```

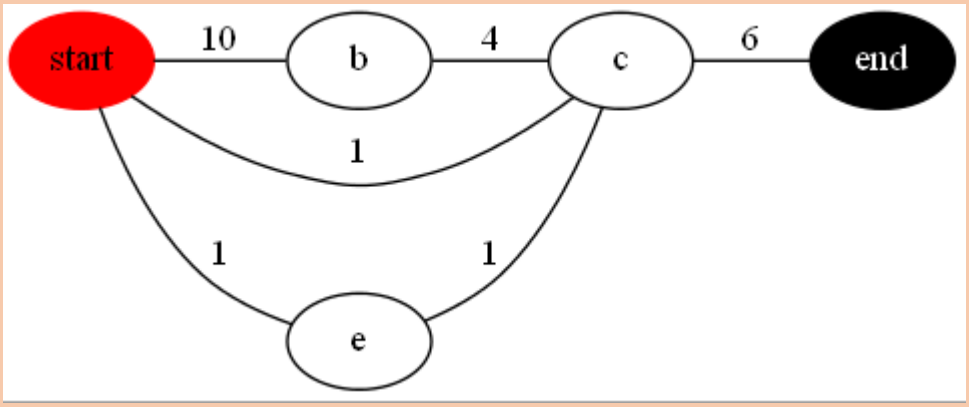


说明：

- **--：** 无向图连线，可指定 节点集合 到 节点集合的连线，实现批量指定连线。
- **rankdir=LR:**指定图形排布方式，**TB:**从上到下，**BT:**从下到上；**LR:**从左到右；**RL:**从右到左。默认为 TB
- **位置：**图形和连线出现的位置与图形元素的声明顺序有关

1.2 带标签加权无向图

```
graph {
    rankdir=LR
    a [label="start",color=red,style=filled]
    d [label="end",color=black,fontcolor=white,style=filled]

    a -- b[label="10",weight="10"];
    a -- c[label="1",weight="1"];
    a -- e[label="1",weight="1"];

    b -- c[label="4",weight="4"];
    e -- c[label="1",weight="1"];

    c -- d[label="6",weight="6"];
}
```

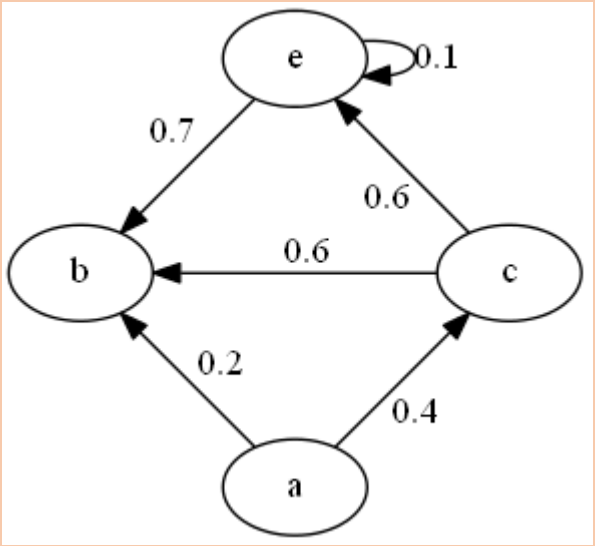


说明：

- label="start":指定节点或连线的标签内容
- color=red:指定节点的颜色
- fontcolor=white:指定节点标签字体的颜色
- style=filled:指定 节点或连线 风格，filled:填充满，对于线:style=dotted:虚线
- weight:两个节点之间的连线 weight 越大，则节点靠得更近。

1.3 有向图

```
digraph {
    a -> b[label="0.2",weight="2"];
    a -> c[label="0.4",weight="4"];
    c -> b[label="0.6",weight="6"];
    c -> e[label="0.6",weight="6"];
    e -> e[label="0.1",weight="1"];
    e -> b[label="0.7",weight="7"];
}
```



说明：

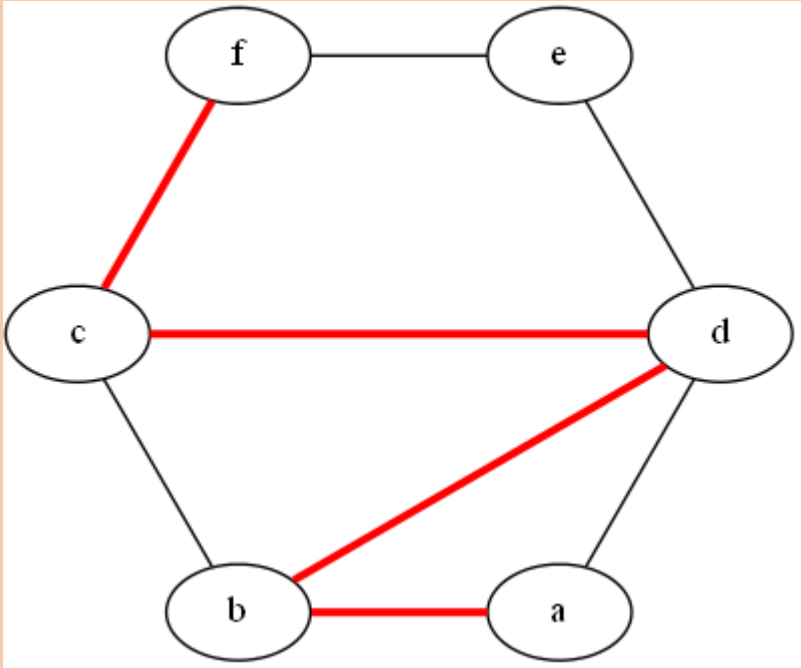
画图引擎：要画出右图，需要切换画图引擎为 circo.

1.3.1 画图引擎

图形引擎	特点
circo	适合多环路结构的图形
dot	
neato	
fdp	
sfdp	
twopi	放射状布局

1.4 标注路径

由 circo 图形引擎生成：

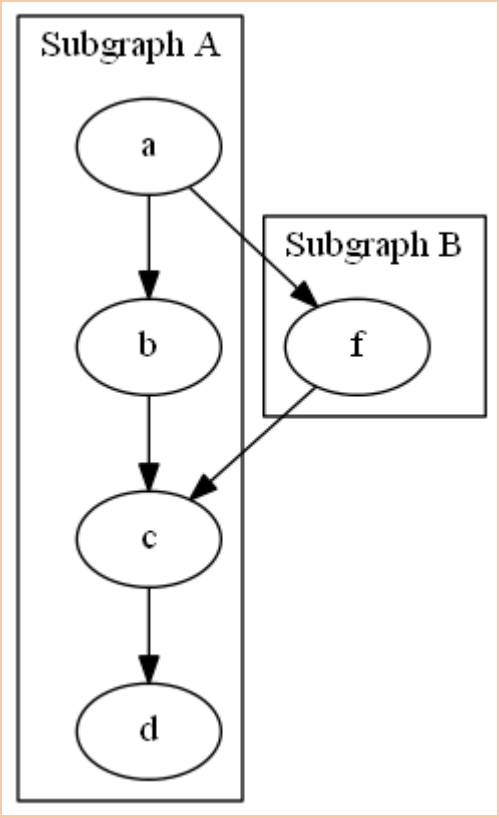



```
graph {
    a -- b -- d -- c -- f[color=red,penwidth=3.0];
    b -- c;
    d -- e;
    e -- f;
    a -- d;
}
```

1.5 子图

由 dot 图形引擎生成：

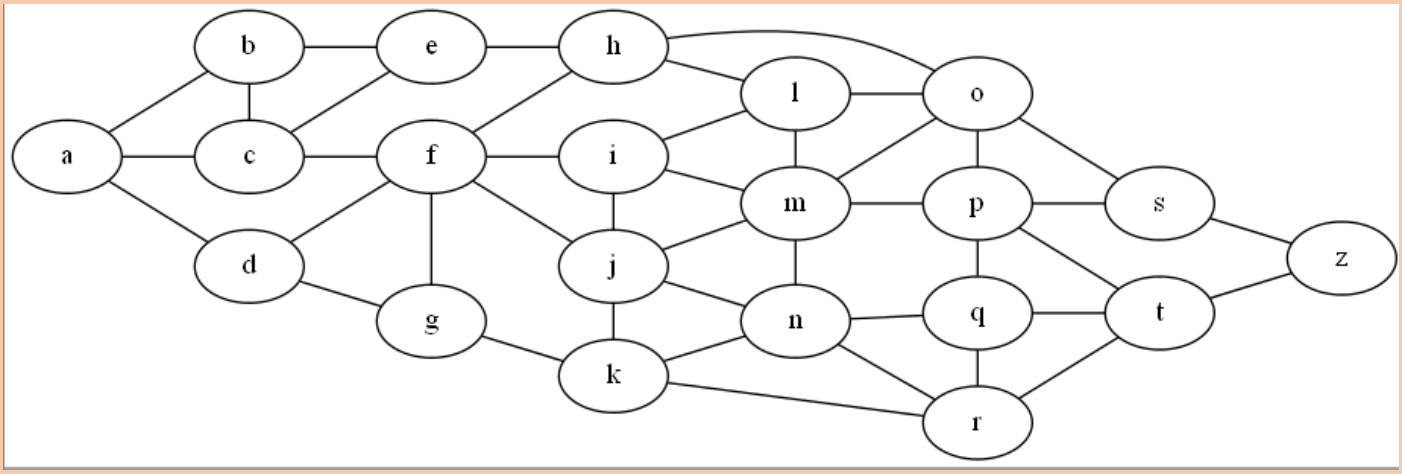
```
digraph {
    splines=line;
    subgraph cluster_0 {
       label="Subgraph A";
       a -> b;
       b -> c;
       c -> d;
    }

    subgraph cluster_1 {
       label="Subgraph B";
       a -> f;
       f -> c;
    }
}
```



说明：

- splines=line：指定只使用直线
- subgraph clusterXXX：子图命名必需以 cluster 开关，否则无法合并到一个框图中。而且只有 dot 引擎支持。

1.6 大型图形:rank=same 对齐




```
graph {
   rankdir=LR;
   a -- { b c d }; b -- { c e }; c -- { e f }; d -- { f g }; e -- h;
   f -- { h i j g }; g -- k; h -- { o l }; i -- { l m j }; j -- { m n k };
   k -- { n r }; l -- { o m }; m -- { o p n }; n -- { q r };
   o -- { s p }; p -- { s t q }; q -- { t r }; r -- t; s -- z; t -- z;
   { rank=same b c d };
   { rank=same e f g } ;
   { rank=same h i j k };
```

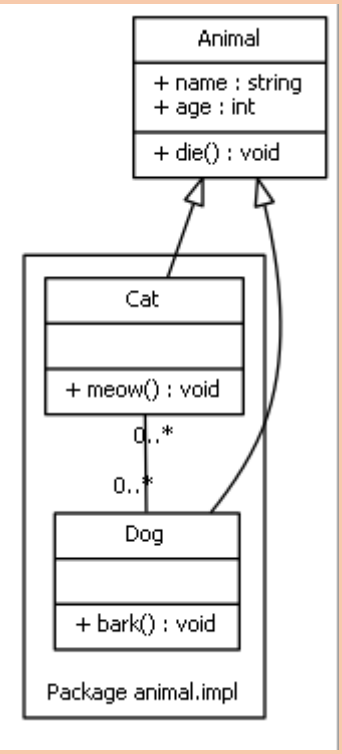
```
    { rank=same l m n };
    { rank=same o p q r };
    { rank=same s t };
}
```

说明：

- a -- { b c d }; ： 指定 节点 -> 节点集合 的连线
- rank=same：将节点对齐排列，左右或上下
- ranksep=1：指定两级 rank 之间的距离,inch. 在上图中 ranksep 越大，a,c 离的越开
- nodesep=1：指定同级 rank 之间的距离,inch. 在上图中 nodesep 越大,a,c,d 离的越开

1.7 UML 元素

引用：http://www.ffnn.nl/pages/articles/media/uml-diagrams-using-graphviz-dot.php
使用 dot 引擎生成：如果使用 circo 引擎，则无法生成子图




```
digraph G {
	fontname = "Bitstream Vera Sans"
	fontsize = 8

	node [
		fontname = "Bitstream Vera Sans"
		fontsize = 8
		shape = "record"
	]

	edge [
		fontname = "Bitstream Vera Sans"
		fontsize = 8
	]
```

```
	Animal [
		label = "{Animal|+ name : string\l+ age : int\l|+ die() : void\l}"
	]

	subgraph clusterAnimalImpl {
		label = "Package animal.impl"

		Dog [
			label = "{Dog||+ bark() : void\l}"
		]

		Cat [
			label = "{Cat||+ meow() : void\l}"
		]
	}

	edge [
		arrowhead = "empty"
	]

	Dog -> Animal
	Cat -> Animal

	edge [
		arrowhead = "none"

		headlabel = "0..*"
		taillabel = "0..*"
	]

	Dog -> Cat
}
```

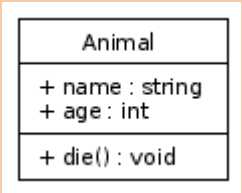
1.7.1 node[..],edge[..]设置节点、连线属性

1.7.2 node[shape=”record”]

设置节点为 record，这样的节点可以被分割，适合构造类图

1.7.3 类表示：Animal Class

```
Animal [
	label = "{Animal|+ name : string\l+ age : int\l|+ die() : void\l}"
]
```

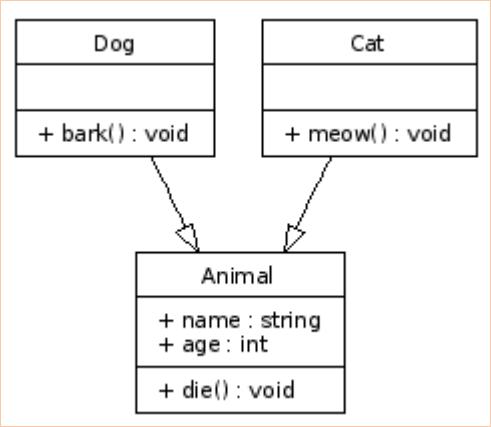


说明：

- "{" and "}"：表示要创建一个 record 的图形，并带有分隔线。
- "|" ： 代表分隔线。这时用于分隔类名、方法、属性
- "\l" ： 换行，后面的字符左对齐

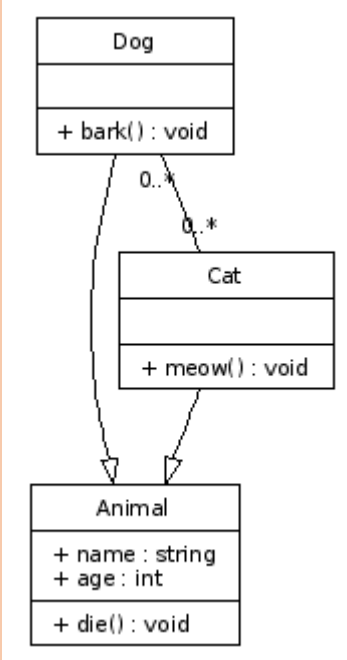
1.7.4 继承关系:edge[arrowhead = "empty"]

```
edge [
        arrowhead = "empty"
]

Dog -> Animal
Cat -> Animal
```



1.7.5 N:M 关系:edge[arrowhead="none",headlabel="",taillabel=""]

```
edge [
    arrowhead = "none"

    headlabel = "0..*"
    taillabel = "0..*"
]
```



1.7.6 包：使用子图实现 subgraph clusterxxx {}

```
subgraph clusterAnimalImpl {
              label = "Package animal.impl"
              ... Cat/Dog 类
}
```