An excerpt from

The Graphviz Cookbook

Rod Waldhoff
rwaldhoff@gmail.com
http://heyrod.com/

ABOUT THIS BOOK

The Graphviz Cookbook, like a regular cookbook, is meant to be a practical guide that *shows you how to create something tangible* and, hopefully, *teaches you how to improvise your own creations* using similar techniques.

The book is organized into four parts:

Part 1: Getting Started introduces the Graphviz tool suite and provides "quick start" instructions to help you get up-and-running with Graphviz for the first time.

Part 2: Ingredients describes the elements of the Graphviz ecosystem in more detail, including an in-depth review of each application in the Graphviz family.

Part 3: Techniques reviews several idioms or "patterns" that crop up often when working with Graphviz such as how to tweak a graph's layout or add a "legend" to a graph. You might think of these as "micro-recipes" that are used again and again.

Part 4: Recipes contains detailed walk-throughs of how to accomplish specific tasks with Graphviz, such as how to spider a web-site to generate a sitemap or how to generate UML diagrams from source files.

Chapter 17

Styling Edges

17.1 Arrowheads (and Arrowtails)

Recall that Graphviz uses the term *tail* to refer to the beginning of a directed edge, and the term *head* to refer to the end. That is, in the edge defined by `A->B`, *node A* is at the tail of the edge and *node B* is at the head (illustrated in Figure 17.1).

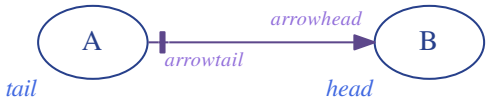

Figure 17.1: *A directed edge begins at the* tail *node and ends at the* head*. An* arrowhead *or* arrowtail *can be used to decorate the head and tail ends of the edge, respectively.*

The direction of an edge in a digraph is typically illustrated by an *arrowhead* that appears at the point at which the edge is attached to the head node.

Graphviz supports attributes that control the placement, direction and shape of the arrows that decorate an edge.

The `dir` (direction) attribute.

By default, layout engines such as `dot` and `neato` render an undirected edge without any arrows at all, and a directed edge with a triangular arrowhead where the edge touches the head node

The `dir` edge attribute can be used to override this default behavior, as illustrated in Figure 17.2.

Specifically, there are five possible values for the `dir` attribute:

1. The default (no `dir` attribute, or an empty string value), which renders an arrowhead on directed edges and no arrows on un-directed edges.

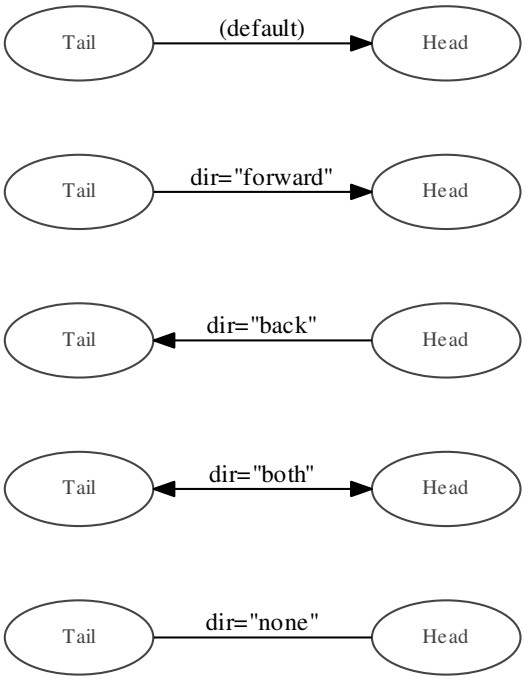

Figure 17.2: *The* `dir` *attribute deterimes where the arrow decorations, if any, are placed on an edge.*

2. `forward`, which renders an arrowhead.
3. `back`, which renders an arrow*tail*.
4. `both`, which renders both an arrowhead and an arrowtail.
5. `none`, which renders neither an arrowhead nor an arrowtail (even on directed edges).

Arrow Types (`arrowhead`, `arrowtail`)

Arrow shapes are controlled by the `arrowhead` and `arrowtail` attributes.

```
digraph G {
  rankdir=LR
  Tail -> Head [dir=both
                arrowhead=tee
                arrowtail=dot]
}
```

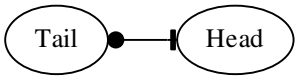

Figure 17.3: *Arrow shapes are controlled by the* `arrowhead` *and* `arrowtail` *attributes.*

Modern versions of Graphviz offer a rich selection of literally *millions* of distinct

arrowheads (and tails) that can be assigned to edges within a digraph. The scheme requires a little explanation.

(We'll use the `arrowhead` attribute in the following discussion, but, when the `dir` attribute is et to `back` or `both`, the exact same syntax can be used with the `arrowtail` attribute.)

There are ten basic arrow shapes—`box`, `crow`, `curved`, `diamond`, `dot`, `inv`, `none`, `normal`, `tee` and `vee`—as illustrated in Figure 17.4.

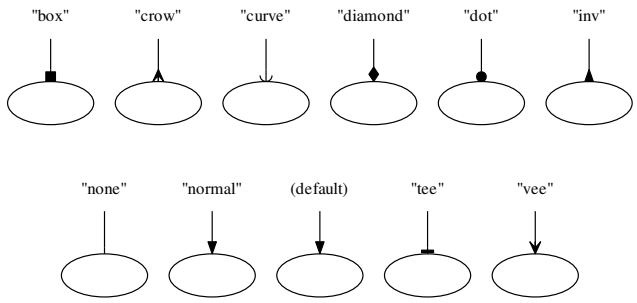

Figure 17.4: *Graphviz has ten basic* `arrowhead` *and* `arrowtail` *shapes. (Note that* `normal` *is the same as the* default *value.)*

By prefixing one of `o`, `r`, `or`, `l` or `ol` to the name of any of the basic arrow shapes one can create up to six variations of each of the ten basic shapes.

As illustrated in Figure 17.5:

1. When the letter `l` or the letter `r` is prefixed to the name of an arrow shape, only the left (or right, respectively) side of the shape will be drawn.
2. When the letter `o` is prefixed to the name of an arrow shape (including the `l`- and `r`-prefixed half-shapes), an "open" or "outline" variation of the arrow will be drawn.

Altogether, there are nearly 60 different arrow shapes that can be formed using the six variations of ten basic shapes.[1] Figure 17.6 contains a chart of all 60 basic and modified arrow shapes.

As if that were not enough, Graphviz offers yet another way to customize arrow shapes. Up to four arrows can be "stacked" together to create a combined arrow, as illustrated in Figure 17.7.

[1] There 60 different *names* for arrow shapes, but fewer than 60 different *shapes* in practice. For instance, the `none` shape only has one variation and the `vee`, `tee` and `curve` shapes have left (`l`) and right (`r`) versions, but no open variations (`o`, `ol` or `or`).

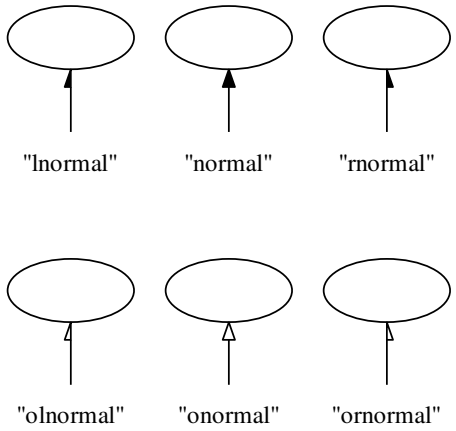

Figure 17.5: *By prefixing one of* `o`, `r`, `or`, `l` *or* `ol` *to one of the basic shape names, one can create up to six variations of each of the basic arrow shapes.*

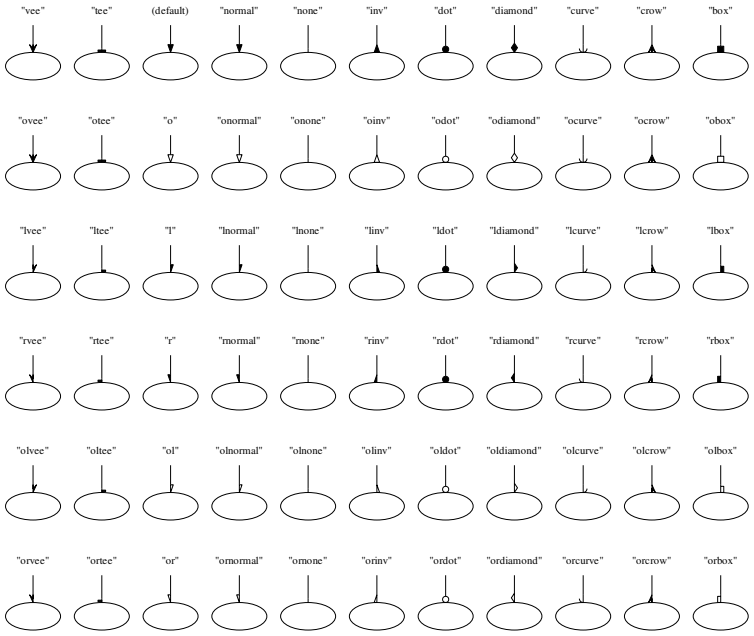

Figure 17.6: *The Graphviz arrow shapes.*

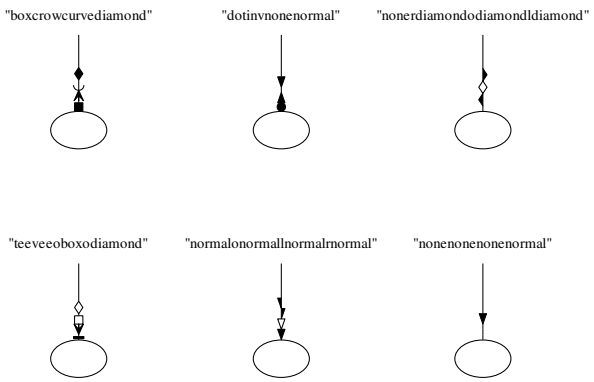

Figure 17.7: *Up to four arrow shape names can be combined to create "stacked" arrow shapes.*

To specify a stack of shapes, simply smush together (concatenate) the names of up to four shapes (without spaces or any other delimiter) and use that combined string as the value of the `arrowhead` attribute. For example, the name `odotcrowloboxnormal` specifies a stack of four arrows: `odot`, `crow`, `lobox` and `normal`. Note that you don't need to specify all four values (they'll default to `none`) and, as illustrated in Figure 17.7, one can use the `none` shape to create some space between the arrow and the node that it points to.

The `arrowsize` attribute

The `arrowsize` attribute specifies the multiplicative factor by which the size of the arrowhead and arrowtail shapes are scaled. The default value is `1.0`. An `arrowsize` of `0.5` will yield shapes that are one-half the default size. An `arrowsize` of `2.0` will yield shapes that are twice the default size. (See Figure 17.8.)

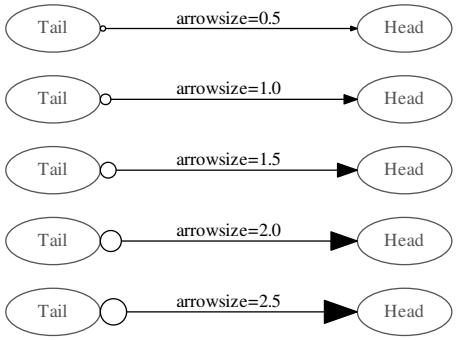

Figure 17.8: *The* `arrowsize` *attribute specifies the factor by which the size of the arrowhead and arrowtail shapes are scaled.*

Currently there is no way to control the size of the arrowhead and arrowtail shapes independently. `arrowsize` controls both.

17.2 Line Style

By default, layout engines such as `dot` and `neato` render each edge as a solid, black line that is one pixel wide.

A handful of attributes can change the way edges are drawn.

The `style` attribute

Graphviz recognizes five basic edge styles: `solid`, `dashed`, `dotted`, `tapered` and `invis` (invisible). As illustrated in Figure 17.9, the keyword `bold` can be added to the style description to produce four more styles.

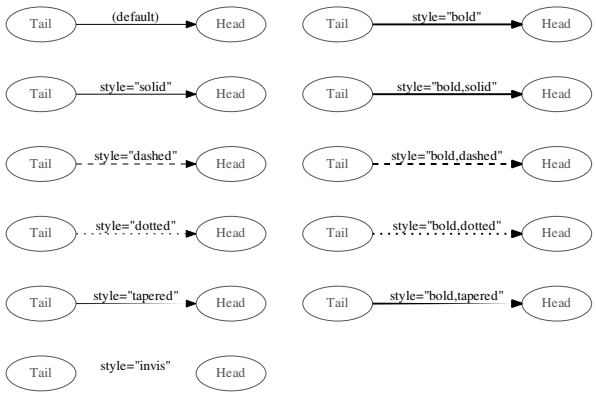

Figure 17.9: *Graphviz recognizes five basic edge* `styles`*:* `solid`*,* `dashed`*,* `dotted`*,* `tapered` *and* `invis` *(invisible).*

The `color` attribute

The `color` attribute is used to change the color of the edge line itself and any arrows with which it has been decorated. It does not alter the color of the edge's `label` or `xlabel` (if any).

The `penwidth` attribute

The width (thickness) of an edge is specified (in points) by the `penwidth` attribute. The default `penwidth` is `1`. The `style=bold` attribute is roughly equivalent to setting the `penwidth` to `2`.

`penwidth` also influences the `arrowhead` and `arrowtail` rendering.

Figure 17.10: *The* `color` *edge attribute sets the color of the line and any arrows with which it has been decorated*

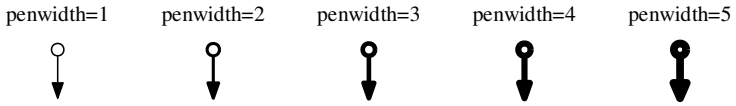

Figure 17.11: *The* `penwidth` *edge attribute specifies the thickness of the edge line and arrows.*

17.3 Labels

Most of the Graphviz layout engines allow one to assign one or more *labels* to decorate an edge.

Label Types

There are several different label attributes, each representing a slightly different position on the edge.

`label` — specifies text that is centered along (and is slightly offset from) the edge's length.

`taillabel` — specifies text that is placed near the tail (beginning) of an edge.

`headlabel` — specifies text that is placed near the head (end) of an edge.

`xlabel` — specifies an *external* label. This is roughly synoymous with `label`, and the `xlabel` is generally placed *near* the center of the edge, but see Section 17.3 for a brief dicussion of the small but important difference between the two.

Figure 17.12 demonstrates the default placement of the `label`, `taillabel`, `headlabel` and `xlabel` text along a few edges.

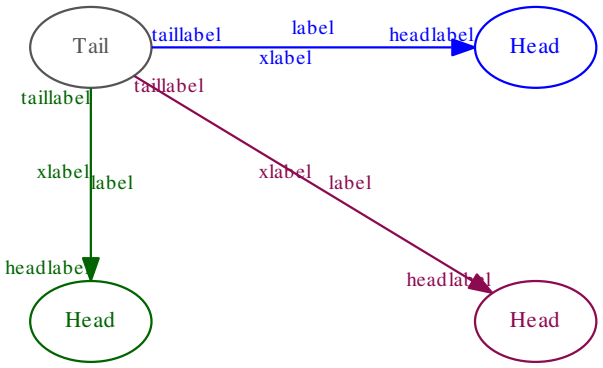

Figure 17.12: *Graphviz has four types of edge label:* `label`, `taillabel`, `headlabel` *and* `xlabel`.

`label` vs. `xlabel`

`dot` and other layout engines take the `label` text into account as the nodes and edges are being laid out. `dot` will attempt to route around the edge labels.

In contrast, the `xlabel` text is placed *after* the nodes and edges have been laid out. Rather than routing edges and nodes around the text, `dot` will attempt to squeeze the `xlabel` text into the space available after the nodes and edges have been placed.

The `forcelabels` graph attribute controls what happens to the `xlabel` values cannot be placed in the available space.

If the `forcelabels` graph attribute is `false`, `dot` will omit any `xlabels` that cannot be made to fit.

If the `forcelabels` graph attribute is `true`, `dot` will render *all* of the edge `xlabels`, even if the `xlabel` text will overlap with nodes, edges or other objects in the diagram.

The default value for `forcelabels` is `true`. You'll need to set `forcelabels=false` if you want `dot` to omit `xlabels` that don't fit.

Label Position (the `labelangle` and `labeldistance` attributes)

By default, the Graphviz layout engines try to place each `headlabel` or `taillabel` in what seems to be a reasonable position. It does this pretty well, as can be seen in Figure 17.13.

If you prefer, you can *override* Graphviz's default `headlabel` and `taillabel` placement using two edge attributes.

The `labelangle` attribute specifies the angle (in degrees) at which the `headlabel` and `taillabel` text begins, relative to the orientation of the edge itself, as illustrated in Figure 17.14 and Figure 17.15 .

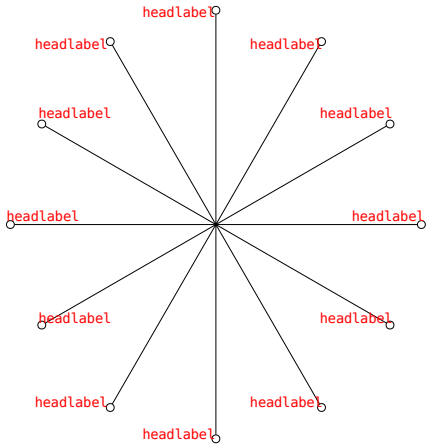

Figure 17.13: *By default, Graphviz tries to place each* `headlabel` *or* `taillabel` *in what seems to be a reasonable position.*

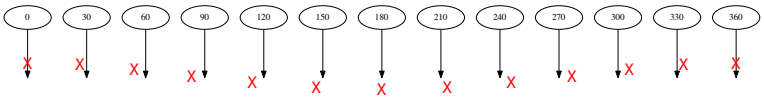

Figure 17.14: *The* `labelangle` *attribute specifies the angle (in degrees) at which the* `headlabel` *and* `taillabel` *text begins, relative to the orientation of the edge itself. If the edge is pointing down,* `labelangle=0` *is at the 12 o'clock position (due north of the endpoint),* `labelangle=90` *is the 9 o'clock position (due west of the endpoint),* `labelangle=180` *is the 6 o'clock position (due south of the endpoint) and* `labelangle=270` *is the 3 o'clock position (due east of the endpoint). (Compare with Figure 17.15.)*

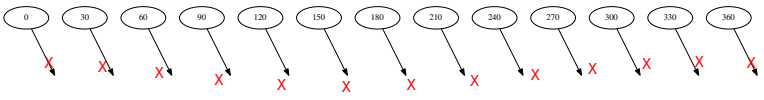

Figure 17.15: *The* `labelangle` *attribute specifies the angle (in degrees) at which the* `headlabel` *and* `taillabel` *text begins, relative to the orientation of the edge itself. (Compare with Figure 17.14).*

The `labeldistance` attribute specifies how close or how far from the edge endpoint the `headlabel` and `taillabel` text is rendered. In particular, the `labeldistance` value specifies the *factor* by which to scale the default distance from the endpoint. That is, `labeldistance=5` doesn't mean to draw the label five *pixels* or *points* away from the endpoint, but five *times* the default distance (of ten points).

Figure 17.16: *The* `labeldistance` *attribute specifies how close or how far from the edge endpoint the* `headlabel` *and* `taillabel` *text is rendered, as a multiplicative factor of the default distance of 10 points.*

Label Font and Color

The `fontname`, `fontsize` and `fontcolor` edge attributes can be used to specify the face, size and color of all of an edge's labels (`label`, `xlabel`, `headlabel` and `taillabel`). (Note that `color` is used to specify the color of an edge's line, and that this is independent of the color used for the labels.)

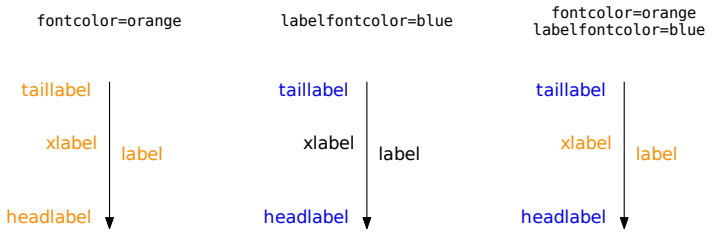

Figure 17.17: `fontname`, `fontsize` *and* `fontcolor` *control the face, size and color of all of an edge's labels. When present,* `labelfontname`, `labelfontsize` *and* `labelfontcolor` *indepedently control the face, size and color of the* `headlabel` *and* `taillabel`.

When present, the `labelfontname`, `labelfontsize` and `labelfontcolor` edge attributes specify the face, size and color of the `headlabel` and `taillabel` text (independent of the `label` or `xlabel` text). Otherwise, the head and tail labels default to the edge's generic `fontname`, `fontsize` and `fontcolor` values.

See Figure 17.17 for a demonstration of these attributes.