NAME

dot – filter for drawing directed graphs
neato – filter for drawing undirected graphs
twopi – filter for radial layouts of graphs
circo – filter for circular layout of graphs
fdp – filter for drawing undirected graphs
sfdp – filter for drawing large undirected graphs
patchwork – filter for squarified tree maps
osage – filter for array-based layouts

SYNOPSIS

dot [*options*] [files]
neato [*options*] [files]
twopi [*options*] [files]
circo [*options*] [files]
fdp [*options*] [files]
sfdp [*options*] [files]
patchwork [*options*] [files]
osage [*options*] [files]

DESCRIPTION

These are a collection of programs for drawing graphs. There is actually only one main program; the specific layout algorithms are implemented as plugins. Thus, they largely share all of the same command-line options.

dot draws directed graphs. It works well on directed acyclic graphs and other graphs that can be drawn as hierarchies or have a natural ‘‘flow.’’

neato draws undirected graphs using a ‘‘spring’’ model and reducing the related energy (see Kamada and Kawai, Information Processing Letters 31:1, April 1989).

twopi draws graphs using a radial layout (see G. Wills, Symposium on Graph Drawing GD’97, September, 1997). Basically, one node is chosen as the center and put at the origin. The remaining nodes are placed on a sequence of concentric circles centered about the origin, each a fixed radial distance from the previous circle. All nodes distance 1 from the center are placed on the first circle; all nodes distance 1 from a node on the first circle are placed on the second circle; and so forth.

circo draws graphs using a circular layout (see Six and Tollis, GD ’99 and ALENEX ’99, and Kaufmann and Wiese, GD ’02.) The tool identifies biconnected components and draws the nodes of the component on a circle. The block-cutpoint tree is then laid out using a recursive radial algorithm. Edge crossings within a circle are minimized by placing as many edges on the circle’s perimeter as possible. In particular, if the component is outerplanar, the component will have a planar layout. If a node belongs to multiple non-trivial biconnected components, the layout puts the node in one of them. By default, this is the first non-trivial component found in the search from the root component.

fdp draws undirected graphs using a ‘‘spring’’ model. It relies on a force-directed approach in the spirit of Fruchterman and Reingold (cf. Software-Practice & Experience 21(11), 1991, pp. 1129-1164).

sfdp also draws undirected graphs using the ‘‘spring’’ model described above, but it uses a multi-scale approach to produce layouts of large graphs in a reasonably short time.

patchwork draws the graph as a squarified treemap (see M. Bruls et al., ‘‘Squarified treemaps’’, Proc. Joint Eurographics and IEEE TCVG Symp. on Visualization, 2000, pp. 33-42). The clusters of the graph are used to specify the tree.

osage draws the graph using its cluster structure. For a given cluster, each of its subclusters is laid out internally. Then the subclusters, plus any remaining nodes, are repositioned based on the cluster’s **pack** and **packmode** attributes.

OUTPUT FORMATS

Graphviz uses an extensible plugin mechanism for its output renderers, so to see what output formats your installation of dot supports you can use ‘‘dot −T:’’ and check the warning message. Also, The plugin mechanism supports multiple implementations of the output formats, allowing variations in the renderers and formatters. To see what variants are available for a particular output format, use, for example: ‘‘dot −Tpng:’’ and to force a particular variant, use, for example: ‘‘dot −Tpng:gd’’

Traditionally, Graphviz supports the following:
−Tdot (Dot format containing layout information),
−Txdot (Dot format containing complete layout information),
−Tps (PostScript),
−Tpdf (PDF),
−Tsvg −Tsvgz (Structured Vector Graphics),
−Tfig (XFIG graphics),
−Tpng (png bitmap graphics),
−Tgif (gif bitmap graphics),
−Tjpg −Tjpeg (jpeg bitmap graphics),
−Tjson (xdot information encoded in JSON),
−Timap (imagemap files for httpd servers for each node or edge that has a non-null **href** attribute.),
−Tcmapx (client-side imagemap for use in html and xhtml).
Additional less common or more special-purpose output formats can be found at https://www.graphviz.org/doc/info/output.html.

Alternative plugins providing support for a given output format can be found from the error message resulting from appending a ’:’ to the format. e.g. **-Tpng:** The first plugin listed is always the default.

The **−P** switch can be used to produce a graph of all output variants supported by plugins in the local installation of graphviz.

GRAPH FILE LANGUAGE

Here is a synopsis of the graph file language, normally using the extension **.gv**, for graphs:

[**strict**] (**graph**|**digraph**) *name* { *statement-list* }
is the top-level graph. If the graph is **strict**, then multiple edges are not allowed between the same pairs of nodes. If it is a directed graph, indicated by **digraph**, then the *edgeop* must be "−>". If it is an undirected **graph** then the *edgeop* must be "−−".

Statements may be:

name=*val*;
node [*name*=*val*];
edge [*name*=*val*];
Set default graph, node, or edge attribute *name* to *val*. Any subgraph, node, or edge appearing after this inherits the new default attributes.

n0 [*name0=val0,name1=val1,...*]; Creates node **n0** (if it does not already exist) and sets its attributes according to the optional list.

n0 *edgeop* n1 *edgeop ... edgeop* nn [*name0=val0,name1=val1,...*];
Creates edges between nodes **n0**, **n1**, ..., **nn** and sets their attributes according to the optional list. Creates nodes as necessary.

[**subgraph** *name*] { *statement-list* }
Creates a subgraph. Subgraphs may be used in place of **n0**, ..., **nn** in the above statements to create edges. [**subgraph** *name*] is optional; if missing, the subgraph is assigned an internal name.

The language accepts both C-style comments /*C...*/ or //...

Attribute names and values are ordinary (C-style) strings. The following sections describe attributes that control graph layout.

A more complete description of the language can be found at https://www.graphviz.org/doc/info/lang.html.

GRAPH, NODE AND EDGE ATTRIBUTES

Graphviz uses the *name=value* attributes, attached to graphs, subgraphs, nodes and edges, to tailor the layout and rendering. We list the more prominent attributes below. The complete list is available at https://www.graphviz.org/doc/info/attrs.html.

Attributes Common to Nodes, Edges, Clusters and Graphs

href=*url* the default url for image map files; in PostScript files, the base URL for all relative URLs, as recognized by Acrobat Distiller 3.0 and up.

URL=*url* (“URL” is a synonym for “href.”)

fontcolor=*colorvalue* sets the label text color.

A *colorvalue* may be "*h,s,v*" (hue, saturation, brightness) floating point numbers between 0 and 1, or an X11 color name such as **white, black, red, green, blue, yellow, magenta,** or **cyan**, or a "*#rrggbb" (red, green, blue, 2 hex characters each) value. See https://www.graphviz.org/doc/info/attrs.html#k:color and https://www.graphviz.org/doc/info/colors.html for further details.*

fontsize=*n* sets the label type size to *n* points.

fontname=*name* sets the label font family name.

label=*text* where *text* may include escaped newlines \n, \l, or \r for center, left, and right justified lines. The string ’\G’ value will be replaced by the graph name. For node labels, the string ’\N’ value will be replaced by the node name. For edges, if the substring ’\T’ is found in a label, it will be replaced by the name of the tail node; if the substring ’\H’ is found in a label, it will be replaced by the name of the head node; if the substring ’\E’ value is found in a label it will be replaced by: *tail_node_name−>head_node_name* or by: *tail_node_name−−head_node_name* for undirected graphs.

Graphviz also supports special HTML-like labels for constructing complex node content. A full-description of these is given at https://www.graphviz.org/doc/info/shapes.html#html.

If a node has **shape=record**, the label may contain recursive box lists delimited by { | }. Port identifiers in labels are set off by angle brackets < >.

Graph Attributes

size="*x,y***"** specifies the maximum bounding box of drawing in inches.

ratio=*f* sets the aspect ratio to *f* which may be a floating point number, or one of the keywords **fill**, **compress**, or **auto**.

layout=*engine* indicates the preferred layout engine (**dot**, **neato**, **fdp**, etc.) overriding the default from the basename of the command or the −K commandline option.

margin=*f* sets the page margin (included in the page size).

ordering=out constrains order of out-edges in a subgraph according to their file sequence.

rotate=90 sets landscape mode. (**orientation=land** is backward compatible but obsolete.)

center=*n* a non-zero value centers the drawing on the page.

color=*colorvalue* sets foreground color (**bgcolor** for background).

overlap=*mode*. This specifies what algorithm should do if any nodes overlap. If mode is **false**, the program uses the Prism algorithm to adjust the nodes to eliminate overlaps. If mode is **scale**, the layout is uniformly scaled up, preserving node sizes, until nodes no longer overlap. The latter technique removes overlaps while preserving symmetry and structure, while the former removes overlaps more compactly but destroys symmetries. If mode is **true** (the default), no repositioning is done. Since the **dot** algorithm always produces a layout with no node overlaps, this attribute is only useful with other layouts.

stylesheet=*"file.css"* includes a reference to a stylesheet in −Tsvg and −Tsvgz outputs. Ignored by other formats.

splines If set to **true**, edges are drawn as splines. If set to **polyline**, edges are drawn as polylines. If set to **ortho**, edges are drawn as orthogonal polylines. In all of these cases, the nodes must not overlap. If

splines=false or **splines=line**, edges are drawn as line segments. The default is **true** for dot, and **false** for all other layouts.

(dot-specific attributes)

nodesep=*f* sets the minimum separation between nodes.

ranksep=*f* sets the minimum separation between ranks.

rankdir=LR|RL|BT requests a left-to-right, right-to-left, or bottom-to-top, drawing.

rank=same (or **min** or **max**) in a subgraph constrains the rank assignment of its nodes. If a subgraph's name has the prefix **cluster**, its nodes are drawn in a distinct rectangle of the layout. Clusters may be nested.

(neato-specific attributes)
mode=*val*. Algorithm for minimizing energy in the layout. By default, **neato** uses stress majorization. If **mode=KK**, it uses a version of gradient descent.

model=*val*. The **neato** model computes the desired distances between all pairs of vertices. By default, it uses the length of the shortest path. If **model** is set to **circuit**, a circuit-resistance model is used. If **model** is set to **subset**, it uses a model whereby the edge length is the number of nodes that are neighbors of exactly one of the edge's vertices.

start=*val*. Requests random initial placement and seeds the random number generator. If *val* is not an integer, the process ID or current time is used as the seed.

epsilon=*n*. Sets the cutoff for the solver. The default is 0.1.

(twopi-specific attributes)
root=*ctr*. This specifies the node to be used as the center of the layout. If not specified, *twopi* will randomly pick one of the nodes that are furthest from a leaf node, where a leaf node is a node of degree 1. If no leaf nodes exists, an arbitrary node is picked as center.

ranksep=*val*. Specifies the radial distance in inches between the sequence of rings. The default is 0.75.

(circo-specific attributes)
root=*nodename*. Specifies the name of a node occurring in the root block. If the graph is disconnected, the **root** node attribute can be used to specify additional root blocks.

mindist=*value*. Sets the minimum separation between all nodes. If not specified then *circo* uses a default value of 1.0.

(fdp-specific attributes)
K=*val*. Sets the default ideal node separation in the layout.

maxiter=*val*. Sets the maximum number of iterations used to layout the graph.

start=*val*. Adjusts the random initial placement of nodes with no specified position. If *val* is an integer, it is used as the seed for the random number generator. If *val* is not an integer, a random system-generated integer, such as the process ID or current time, is used as the seed.

Node Attributes

height=*d* or **width=***d* sets minimum height or width. Adding **fixedsize=true** forces these to be the actual size (text labels are ignored).

shape=*builtin_polygon* record epsf
builtin_polygon can be such values as **plaintext, ellipse, oval, circle, egg, triangle, box, diamond, trapezium, parallelogram, house, hexagon, octagon, note, tab, box3d, or component,**, among others. (Polygons are defined or modified by the following node attributes: **regular**, **peripheries**, **sides**,

orientation, **distortion** and **skew**.) **epsf** uses the node's **shapefile** attribute as the path name of an external EPSF file to be automatically loaded for the node shape.

See https://www.graphviz.org/doc/info/shapes.html for a complete description of node shapes.

color=*colorvalue* sets the outline color, and the default fill color if style=filled and **fillcolor** is not specified.

fillcolor=*colorvalue* sets the fill color when style=filled. If not specified, the fillcolor when style=filled defaults to be the same as the outline color.

style=filled solid dashed dotted bold invis

xlabel=*"text"* specifies a label that will be place near, but outside, of a node. The normal **label** string is placed within the node shape.

target=*"target"* is a target string for client-side imagemaps and SVG, effective when nodes have a URL. The target string is used to determine which window of the browser is used for the URL. Setting it to "_graphviz" will open a new window if it doesn't already exist, or reuse it if it does. If the target string is empty, the default, then no target attribute is included in the output. The substrings '\N' and '\G' are substituted in the same manner as for the node label attribute. Additionally the substring '\L' is substituted with the node label string.

tooltip=*"text"* is a tooltip string for client-side imagemaps and SVG, effective when nodes have a URL. The tooltip string defaults to be the same as the label string, but this attribute permits nodes without labels to still have tooltips thus permitting denser graphs. The substrings '\N' and '\G' are substituted in the same manner as for the node label attribute. Additionally the substring '\L' is substituted with the node label string.

The following attributes apply only to polygon shape nodes:

regular=*n* if *n* is non-zero then the polygon is made regular, i.e. symmetric about the x and y axis, otherwise the polygon takes on the aspect ratio of the label. *builtin_polygons* that are not already regular are made regular by this attribute. *builtin_polygons* that are already regular are not affected (i.e. they cannot be made asymmetric).

peripheries=*n* sets the number of periphery lines drawn around the polygon. This value supersedes the number of periphery lines of *builtin_polygons*.

sides=*n* sets the number of sides to the polygon. *n*<3 results in an ellipse. This attribute is ignored by *builtin_polygons*.

orientation=*f* sets the orientation of the first apex of the polygon counterclockwise from the vertical, in degrees. *f* may be a floating point number. The orientation of labels is not affected by this attribute. This attribute is added to the initial orientation of *builtin_polygons.*

distortion=*f* sets the amount of broadening of the top and narrowing of the bottom of the polygon (relative to its orientation). Floating point values between −1 and +1 are suggested. This attribute is ignored by *builtin_polygons*.

skew=*f* sets the amount of right-displacement of the top and left-displacement of the bottom of the polygon (relative to its orientation). Floating point values between −1 and +1 are suggested. This attribute is ignored by *builtin_polygons*.

(circo-specific attributes)
root=*true/false*. This specifies that the block containing the given node be treated as the root of the spanning tree in the layout.

(neato- and fdp-specific attributes)
pin=*val*. If *val* is **true**, the node will remain at its initial position.

Edge Attributes

weight=*val* where *val* is the cost of the edge. For **dot**, weights must be non-negative integers. Values greater than 1 tend to shorten the edge; weight 0 flat edges are ignored for ordering nodes. In **twopi**, a weight of 0 will cause the edge to be ignored in constructing the underlying spanning tree. For **neato** and **fdp**, a heavier weight will put more emphasis on the algorithm achieving an edge length closer to that specified by the edge's **len** attribute.

style=solid dashed dotted bold invis

color=*colorvalue* sets the line color for edges.

color=*colorvaluelist* a ':' separated list of *colorvalue* creates parallel edges, one edge for each color.

dir=forward back both none controls arrow direction.

tailclip,headclip=false disables endpoint shape clipping.

target=*"text"* is a target string for client-side imagemaps and SVG, effective when edges have a URL. If the target string is empty, the default, then no target attribute is included in the output. The substrings '\T', '\H', '\E' and '\G' are substituted in the same manner as for the edge label attribute. Additionally the substring '\L' is substituted with the edge label string.

tooltip=*"text"* is a tooltip string for client-side imagemaps effective when edges have a URL. The tooltip string defaults to be the same as the edge label string. The substrings '\T', '\H', '\E' and '\G' are substituted in the same manner as for the edge label attribute. Additionally the substring '\L' is substituted with the edge label string.

arrowhead,arrowtail=none, normal, inv, dot, odot, invdot, invodot, tee, empty, invempty, open, halfopen, diamond, odiamond, box, obox, crow. Specifies the shape of the glyph occurring where the edge touches the head or tail node, respectively. Note that this only specifies the shape. The **dir** attribute determines whether or not the glyph is drawn.

arrowsize=*val* specifies a multiplicative scale factor for the size of the arrowhead. inv_length=6,inv_width=7,dot_radius=2)

headlabel,taillabel=*text* for labels appearing near the head and tail nodes of an edge. **labelfontcolor**, **labelfontname**, **labelfontsize** for head and tail labels. The substrings '\T', '\H', '\E' and '\G' are substituted in the same manner as for the edge label attribute. Additionally the substring '\L' is substituted with the edge label string.

headhref=*"url"* sets the url for the head port in imagemap, PostScript and SVG files. The substrings '\T', '\H', '\E' and '\G' are substituted in the same manner as for the edge label attribute. Additionally the substring '\L' is substituted with the edge label string.

headURL=*"url"* (**headURL** is a synonym for **headhref**.)

headtarget=*"headtarget"* is a target string for client-side imagemaps and SVG, effective when edge heads have a URL. The headtarget string is used to determine which window of the browser is used for the URL. If the headtarget string is empty, the default, then headtarget defaults to the same value as target for the edge. The substrings '\T', '\H', '\E' and '\G' are substituted in the same manner as for the edge label attribute. Additionally the substring '\L' is substituted with the edge label string.

headtooltip=*"tooltip"* is a tooltip string for client-side imagemaps effective when head ports have a URL. The tooltip string defaults to be the same as the headlabel string. The substrings '\T', '\H', and '\E' are substituted in the same manner as for the edge label attribute. Additionally the substring '\L' is substituted with the edge label string.

tailhref=*"url"* sets the url for the tail port in imagemap, PostScript and SVG files. The substrings '\T', '\H', '\E' and '\G' are substituted in the same manner as for the edge label attribute. Additionally the substring '\L' is substituted with the edge label string.

tailURL=*"url"* (**tailURL** is a synonym for **tailhref**.)

tailtarget=*"tailtarget"* is a target string for client-side imagemaps and SVG, effective when edge tails have a URL. The tailtarget string is used to determine which window of the browser is used for the URL. If the

tailtarget string is empty, the default, then tailtarget defaults to the same value as target for the edge. The substrings '\T', '\H', '\E' and '\G' are substituted in the same manner as for the edge label attribute. Additionally the substring '\L' is substituted with the edge label string.

tailtooltip=*"tooltip"* is a tooltip string for client-side imagemaps effective when tail ports have a URL. The tooltip string defaults to be the same as the taillabel string. The substrings '\T', '\H', '\E' and '\G' are substituted in the same manner as for the edge label attribute. Additionally the substring '\L' is substituted with the edge label string.

labeldistance and **labelangle** (in degrees CCW) specify the placement of head and tail labels.

decorate draws line from edge to label.

samehead,sametail aim edges having the same value to the same port, using the average landing point.

(dot-specific attributes)
constraint=false causes an edge to be ignored for rank assignment.

minlen=*n* where *n* is an integer factor that applies to the edge length (ranks for normal edges, or minimum node separation for flat edges).

xlabel=*"text"* Edge labels in **dot** are treated as special types of nodes, with space allocated for them during node layout. This can sometimes deform the edge routing. If an **xlabel** is used instead, the label is placed after all nodes and edges have been positioned. In turn, this may mean that there is some overlap among the labels.

(neato and fdp-specific attributes)
len=*f* sets the optimal length of an edge. The default is 1.0.

COMMAND-LINE OPTIONS

−G sets a default graph attribute.
−N sets a default node attribute.
−E sets a default edge attribute. Example: **−Gsize="7,8" −Nshape=box −Efontsize=8**

−l*file* loads custom PostScript library files. Usually these define custom shapes or styles. If **−l** is given by itself, the standard library is omitted.

−T*lang* sets the output language as described above.

−n[**1**|**2**] (no-op) If set, neato assumes nodes have already been positioned and all nodes have a pos attribute giving the positions. It then performs an optional adjustment to remove node-node overlap, depending on the value of the overlap attribute, computes the edge layouts, depending on the value of the **splines** attribute, and emits the graph in the appropriate format. If num is supplied, the following actions occur:
num = 1
Equivalent to −n.
num > 1
Use node positions as specified, with no adjustment to remove node-node overlaps, and use any edge layouts already specified by the pos attribute. neato computes an edge layout for any edge that does not have a **pos** attribute. As usual, edge layout is guided by the **splines** attribute.

−K*layout* override the default layout engine implied by the command name.

−O automatically generate output filenames based on the input filename and the −T format.

−P generate a graph of the currently available plugins.

−v (verbose) prints various information useful for debugging.

−c configure plugins.

−q*level* set level of message suppression. The default is 1.

−s*fscale* scale input by *fscale*, the default is 72.

−y invert y coordinate in output.

−o*file* write output to *file*.

−x reduce graph.

−Lg don't use grid.

−LO use old attractive force.

−Ln*i* set number of iterations to *i*.

−LU*i* set unscaled factor to *i*.

−LC*v* set overlap expansion factor to *v*.

−LT[*]*v* set temperature (temperature factor) to *v*.

−V (version) prints version information and exits.

−? prints the usage and exits.

A complete description of the available command-line options can be found at https://www.graphviz.org/doc/info/command.html.

EXAMPLES

```
digraph test123 {
    a −> b −> c;
    a −> {x y};
    b [shape=box];
    c [label="hello\nworld",color=blue,fontsize=24,
      fontname="Palatino−Italic",fontcolor=red,style=filled];
    a −> z [label="hi", weight=100];
    x −> z [label="multi−line\nlabel"];
    edge [style=dashed,color=red];
    b −> x;
    {rank=same; b x}
}
```

```
graph test123 {
    a −− b −− c;
    a −− {x y};
    x −− c [w=10.0];
    x −− y [w=5.0,len=3];
}
```

CAVEATS

Edge splines can overlap unintentionally.

Flat edge labels are slightly broken. Intercluster edge labels are totally broken.

Because unconstrained optimization is employed, node boxes can possibly overlap or touch unrelated edges. All existing spring embedders seem to have this limitation.

Apparently reasonable attempts to pin nodes or adjust edge lengths and weights can cause instability.

AUTHORS

Stephen C. North <north@research.att.com>
Emden R. Gansner <erg@graphviz.org>
John C. Ellson <ellson@research.att.com>
Yifan Hu <yifanhu@yahoo.com>

The bitmap driver (PNG, GIF etc) is by Thomas Boutell, <http://www.boutell.com/gd>

The Truetype font renderer is from the Freetype Project (David Turner, Robert Wilhelm, and Werner Lemberg) (who can be contacted at freetype−devel@lists.lrz−muenchen.de).

SEE ALSO

This man page contains only a small amount of the information related to the Graphviz layout programs. The most complete information can be found at https://www.graphviz.org/documentation/, especially in the on-line reference pages. Most of these documents are also available in the *doc* and *doc/info* subtrees in the source and binary distributions.

tcldot(n)
xcolors(1)
libcgraph(3)

E. R. Gansner, S. C. North, K. P. Vo, "DAG - A Program to Draw Directed Graphs", Software - Practice and Experience 17(1), 1988, pp. 1047-1062.
E. R. Gansner, E. Koutsofios, S. C. North, K. P. Vo, "A Technique for Drawing Directed Graphs," IEEE Trans. on Soft. Eng. 19(3), 1993, pp. 214-230.
S. North and E. Koutsofios, "Applications of graph visualization", Graphics Interface 94, pp. 234-245.
E. R. Gansner and E. Koutsofios and S. C. North, "Drawing Graphs with dot," Available at https://www.graphviz.org/pdf/dotguide.pdf.
S. C. North, "NEATO User's Manual". Available https://www.graphviz.org/pdf/neatoguide.pdf.
E. R. Gansner and Y. Hu, "Efficient, Proximity-Preserving Node Overlap Removal", J. Graph Algorithms Appl., 14(1) pp. 53-74, 2010.

NOTES

On non-Windows platforms, sending **SIGUSR1** to Graphviz can be used to toggle on and off some extra points reporting in the neatogen code.