

**NAME**

gv\_ruby - graph manipulation in ruby

**SYNOPSIS**

```
#!/usr/bin/ruby
require 'gv'
```

**USAGE****INTRODUCTION**

**gv\_ruby** is a dynamically loaded extension for **ruby** that provides access to the graph facilities of **graphviz**.

**COMMANDS****New graphs**

New empty graph

```
graph_handle Gv.graph (name);
graph_handle Gv.digraph (name);
graph_handle Gv.strictgraph (name);
graph_handle Gv.strictdigraph (name);
```

New graph from a dot-syntax string or file

```
graph_handle Gv.readstring (string);
graph_handle Gv.read (string filename);
graph_handle Gv.read (channel);
```

Add new subgraph to existing graph

```
graph_handle Gv.graph (graph_handle, name);
```

**New nodes**

Add new node to existing graph

```
node_handle Gv.node (graph_handle, name);
```

**New edges**

Add new edge between existing nodes

```
edge_handle Gv.edge (tail_node_handle, head_node_handle);
```

Add a new edge between an existing tail node, and a named head node which will be induced in the graph if it doesn't already exist

```
edge_handle Gv.edge (tail_node_handle, head_name);
```

Add a new edge between an existing head node, and a named tail node which will be induced in the graph if it doesn't already exist

```
edge_handle Gv.edge (tail_name, head_node_handle);
```

Add a new edge between named tail and head nodes which will be induced in the graph if they don't already exist

```
edge_handle Gv.edge (graph_handle, tail_name, head_name);
```

**Setting attribute values**

Set value of named attribute of graph/node/edge - creating attribute if necessary

```
string Gv.setv (graph_handle, attr_name, attr_value);
string Gv.setv (node_handle, attr_name, attr_value);
string Gv.setv (edge_handle, attr_name, attr_value);
```

Set value of existing attribute of graph/node/edge (using attribute handle)

```
string Gv.setv (graph_handle, attr_handle, attr_value);
string Gv.setv (node_handle, attr_handle, attr_value);
```

```
string Gv.setv (edge_handle, attr_handle, attr_value);
```

### Getting attribute values

Get value of named attribute of graph/node/edge

```
string Gv.getv (graph_handle, attr_name);
string Gv.getv (node_handle, attr_name);
string Gv.getv (edge_handle, attr_name);
```

Get value of attribute of graph/node/edge (using attribute handle)

```
string Gv.getv (graph_handle, attr_handle);
string Gv.getv (node_handle, attr_handle);
string Gv.getv (edge_handle, attr_handle);
```

### Obtain names from handles

```
string Gv.nameof (graph_handle);
string Gv.nameof (node_handle);
string Gv.nameof (attr_handle);
```

### Find handles from names

```
graph_handle Gv.findsubg (graph_handle, name);
node_handle Gv.findnode (graph_handle, name);
edge_handle Gv.findedge (tail_node_handle, head_node_handle);
attribute_handle Gv.findattr (graph_handle, name);
attribute_handle Gv.findattr (node_handle, name);
attribute_handle Gv.findattr (edge_handle, name);
```

### Misc graph navigators returning handles

```
node_handle Gv.headof (edge_handle);
node_handle Gv.tailof (edge_handle);
graph_handle Gv.graphof (graph_handle);
graph_handle Gv.graphof (edge_handle);
graph_handle Gv.graphof (node_handle);
graph_handle Gv.rootof (graph_handle);
```

### Obtain handles of proto node/edge for setting default attribute values

```
node_handle Gv.protonode (graph_handle);
edge_handle Gv.protoedge (graph_handle);
```

### Iterators

Iteration termination tests

```
bool Gv.ok (graph_handle);
bool Gv.ok (node_handle);
bool Gv.ok (edge_handle);
bool Gv.ok (attr_handle);
```

Iterate over subgraphs of a graph

```
graph_handle Gv.firstsubg (graph_handle);
graph_handle Gv.nextsubg (graph_handle, subgraph_handle);
```

Iterate over supergraphs of a graph (obscure and rarely useful)

```
graph_handle Gv.firstsupg (graph_handle);
graph_handle Gv.nextsupg (graph_handle, subgraph_handle);
```

Iterate over edges of a graph

```
edge_handle Gv.firstedge (graph_handle);
edge_handle Gv.nextedge (graph_handle, edge_handle);
```

Iterate over outedges of a graph

```
edge_handle Gv.firstout (graph_handle);  
edge_handle Gv.nextout (graph_handle, edge_handle);
```

Iterate over edges of a node

```
edge_handle Gv.firstedge (node_handle);  
edge_handle Gv.nextedge (node_handle, edge_handle);
```

Iterate over out-edges of a node

```
edge_handle Gv.firstout (node_handle);  
edge_handle Gv.nextout (node_handle, edge_handle);
```

Iterate over head nodes reachable from out-edges of a node

```
node_handle Gv.firsthead (node_handle);  
node_handle Gv.nexthead (node_handle, head_node_handle);
```

Iterate over in-edges of a graph

```
edge_handle Gv.firstin (graph_handle);  
edge_handle Gv.nextin (node_handle, edge_handle);
```

Iterate over in-edges of a node

```
edge_handle Gv.firstin (node_handle);  
edge_handle Gv.nextin (graph_handle, edge_handle);
```

Iterate over tail nodes reachable from in-edges of a node

```
node_handle Gv.firsttail (node_handle);  
node_handle Gv.nexttail (node_handle, tail_node_handle);
```

Iterate over nodes of a graph

```
node_handle Gv.firstnode (graph_handle);  
node_handle Gv.nextnode (graph_handle, node_handle);
```

Iterate over nodes of an edge

```
node_handle Gv.firstnode (edge_handle);  
node_handle Gv.nextnode (edge_handle, node_handle);
```

Iterate over attributes of a graph

```
attribute_handle Gv.firstattr (graph_handle);  
attribute_handle Gv.nextattr (graph_handle, attr_handle);
```

Iterate over attributes of an edge

```
attribute_handle Gv.firstattr (edge_handle);  
attribute_handle Gv.nextattr (edge_handle, attr_handle);
```

Iterate over attributes of a node

```
attribute_handle Gv.firstattr (node_handle);  
attribute_handle Gv.nextattr (node_handle, attr_handle);
```

### Remove graph objects

```
bool Gv.rm (graph_handle);  
bool Gv.rm (node_handle);  
bool Gv.rm (edge_handle);
```

### Layout

Annotate a graph with layout attributes and values using a specific layout engine

```
bool Gv.layout (graph_handle, string engine);
```

### Render

Render a layout into attributes of the graph

```
bool Gv.render (graph_handle);
```

Render a layout to stdout

*bool* **Gv.render** (*graph\_handle*, *string format*);

Render to an open file

*bool* **Gv.render** (*graph\_handle*, *string format*, *channel fout*);

Render a layout to an unopened file by name

*bool* **Gv.render** (*graph\_handle*, *string format*, *string filename*);

Render to a string result

*string* **Gv.renderresult** (*graph\_handle ing*, *string format*);

**Gv.renderresult** (*graph\_handle*, *string format*, *string outdata*);

Render to an open channel

*bool* **Gv.renderchannel** (*graph\_handle*, *string format*, *string channelname*);

Render a layout to a malloc'ed string, to be free'd by the caller

(deprecated - too easy to leak memory)

(still needed for "eval [gv::renderdata \$G tk]" )

*string* **Gv.renderdata** (*graph\_handle*, *string format*);

Writing graph back to file

*bool* **Gv.write** (*graph\_handle*, *string filename*);

*bool* **Gv.write** (*graph\_handle*, *channel*);

Graph transformation tools

*bool* **Gv.tred** (*graph\_handle*);

## KEYWORDS

graph, dot, neato, fdp, circo, twopi, ruby.