NAME
>   gv_sharp - graph manipulation in sharp

SYNOPSIS
USAGE
INTRODUCTION
>   **gv_sharp** is a dynamically loaded extension for **sharp** that provides access to the graph facilities of **graphviz.**

COMMANDS
>   **New graphs**

>   New empty graph
>   >   *SWIGTYPE_p_Agraph_t* **gv.graph** *(string name);*
>   >   *SWIGTYPE_p_Agraph_t* **gv.digraph** *(string name);*
>   >   *SWIGTYPE_p_Agraph_t* **gv.strictgraph** *(string name);*
>   >   *SWIGTYPE_p_Agraph_t* **gv.strictdigraph** *(string name);*

>   New graph from a dot-syntax string or file
>   >   *SWIGTYPE_p_Agraph_t* **gv.readstring** *(string string);*
>   >   *SWIGTYPE_p_Agraph_t* **gv.read** *(string filename);*
>   >   *SWIGTYPE_p_Agraph_t* **gv.read** *(SWIGTYPE_p_FILE f);*

>   Add new subgraph to existing graph
>   >   *SWIGTYPE_p_Agraph_t* **gv.graph** *(SWIGTYPE_p_Agraph_t g, string name);*

>   **New nodes**

>   Add new node to existing graph
>   >   *SWIGTYPE_p_Agnode_t* **gv.node** *(SWIGTYPE_p_Agraph_t g, string name);*

>   **New edges**

>   Add new edge between existing nodes
>   >   *SWIGTYPE_p_Agedge_t* **gv.edge** *(SWIGTYPE_p_Agnode_t t, SWIGTYPE_p_Agnode_t h);*

>   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
>   >   *SWIGTYPE_p_Agedge_t* **gv.edge** *(SWIGTYPE_p_Agnode_t t, string hname);*

>   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
>   >   *SWIGTYPE_p_Agedge_t* **gv.edge** *(string tname, SWIGTYPE_p_Agnode_t h);*

>   Add a new edge between named tail and head nodes which will be induced in the graph if they don't already exist
>   >   *SWIGTYPE_p_Agedge_t* **gv.edge** *(SWIGTYPE_p_Agraph_t g, string tname, string hname);*

>   **Setting attribute values**

>   Set value of named attribute of graph/node/edge - creating attribute if necessary
>   >   *string* **gv.setv** *(SWIGTYPE_p_Agraph_t g, string attr, string val);*
>   >   *string* **gv.setv** *(SWIGTYPE_p_Agnode_t n, string attr, string val);*
>   >   *string* **gv.setv** *(SWIGTYPE_p_Agedge_t e, string attr, string val);*

>   Set value of existing attribute of graph/node/edge (using attribute handle)
>   >   *string* **gv.setv** *(SWIGTYPE_p_Agraph_t g, SWIGTYPE_p_Agsym_t a, string val);*
>   >   *string* **gv.setv** *(SWIGTYPE_p_Agnode_t n, SWIGTYPE_p_Agsym_t a, string val);*
>   >   *string* **gv.setv** *(SWIGTYPE_p_Agedge_t e, SWIGTYPE_p_Agsym_t a, string val);*

Getting attribute values

Get value of named attribute of graph/node/edge
> *string* **gv.getv** *(SWIGTYPE_p_Agraph_t g, string attr);*
> *string* **gv.getv** *(SWIGTYPE_p_Agnode_t n, string attr);*
> *string* **gv.getv** *(SWIGTYPE_p_Agedge_t e, string attr);*

Get value of attribute of graph/node/edge (using attribute handle)
> *string* **gv.getv** *(SWIGTYPE_p_Agraph_t g, SWIGTYPE_p_Agsym_t a);*
> *string* **gv.getv** *(SWIGTYPE_p_Agnode_t n, SWIGTYPE_p_Agsym_t a);*
> *string* **gv.getv** *(SWIGTYPE_p_Agedge_t e, SWIGTYPE_p_Agsym_t a);*

Obtain names from handles
> *string* **gv.nameof** *(SWIGTYPE_p_Agraph_t g);*
> *string* **gv.nameof** *(SWIGTYPE_p_Agnode_t n);*
> *string* **gv.nameof** *(SWIGTYPE_p_Agsym_t a);*

Find handles from names
> *SWIGTYPE_p_Agraph_t* **gv.findsubg** *(SWIGTYPE_p_Agraph_t g, string name);*
> *SWIGTYPE_p_Agnode_t* **gv.findnode** *(SWIGTYPE_p_Agraph_t g, string name);*
> *SWIGTYPE_p_Agedge_t* **gv.findedge** *(SWIGTYPE_p_Agnode_t t, SWIGTYPE_p_Agnode_t h);*
> *SWIGTYPE_p_Agsym_t* **gv.findattr** *(SWIGTYPE_p_Agraph_t g, string name);*
> *SWIGTYPE_p_Agsym_t* **gv.findattr** *(SWIGTYPE_p_Agnode_t n, string name);*
> *SWIGTYPE_p_Agsym_t* **gv.findattr** *(SWIGTYPE_p_Agedge_t e, string name);*

Misc graph navigators returning handles
> *SWIGTYPE_p_Agnode_t* **gv.headof** *(SWIGTYPE_p_Agedge_t e);*
> *SWIGTYPE_p_Agnode_t* **gv.tailof** *(SWIGTYPE_p_Agedge_t e);*
> *SWIGTYPE_p_Agraph_t* **gv.graphof** *(SWIGTYPE_p_Agraph_t g);*
> *SWIGTYPE_p_Agraph_t* **gv.graphof** *(SWIGTYPE_p_Agedge_t e);*
> *SWIGTYPE_p_Agraph_t* **gv.graphof** *(SWIGTYPE_p_Agnode_t n);*
> *SWIGTYPE_p_Agraph_t* **gv.rootof** *(SWIGTYPE_p_Agraph_t g);*

Obtain handles of proto node/edge for setting default attribute values
> *SWIGTYPE_p_Agnode_t* **gv.protonode** *(SWIGTYPE_p_Agraph_t g);*
> *SWIGTYPE_p_Agedge_t* **gv.protoedge** *(SWIGTYPE_p_Agraph_t g);*

Iterators

Iteration termination tests
> *bool* **gv.ok** *(SWIGTYPE_p_Agraph_t g);*
> *bool* **gv.ok** *(SWIGTYPE_p_Agnode_t n);*
> *bool* **gv.ok** *(SWIGTYPE_p_Agedge_t e);*
> *bool* **gv.ok** *(SWIGTYPE_p_Agsym_t a);*

Iterate over subgraphs of a graph
> *SWIGTYPE_p_Agraph_t* **gv.firstsubg** *(SWIGTYPE_p_Agraph_t g);*
> *SWIGTYPE_p_Agraph_t* **gv.nextsubg** *(SWIGTYPE_p_Agraph_t g, SWIGTYPE_p_Agraph_t sg);*

Iterate over supergraphs of a graph (obscure and rarely useful)
> *SWIGTYPE_p_Agraph_t* **gv.firstsupg** *(SWIGTYPE_p_Agraph_t g);*
> *SWIGTYPE_p_Agraph_t* **gv.nextsupg** *(SWIGTYPE_p_Agraph_t g, SWIGTYPE_p_Agraph_t sg);*

Iterate over edges of a graph
> *SWIGTYPE_p_Agedge_t* **gv.firstedge** *(SWIGTYPE_p_Agraph_t g);*
> *SWIGTYPE_p_Agedge_t* **gv.nextedge** *(SWIGTYPE_p_Agraph_t g, SWIGTYPE_p_Agedge_t e);*

Iterate over outedges of a graph
> *SWIGTYPE_p_Agedge_t* **gv.firstout** *(SWIGTYPE_p_Agraph_t g);*
> *SWIGTYPE_p_Agedge_t* **gv.nextout** *(SWIGTYPE_p_Agraph_t g, SWIGTYPE_p_Agedge_t e);*

Iterate over edges of a node
   SWIGTYPE_p_Agedge_t **gv.firstedge** *(SWIGTYPE_p_Agnode_t n);*
   SWIGTYPE_p_Agedge_t **gv.nextedge** *(SWIGTYPE_p_Agnode_t n, SWIGTYPE_p_Agedge_t e);*

Iterate over out-edges of a node
   SWIGTYPE_p_Agedge_t **gv.firstout** *(SWIGTYPE_p_Agnode_t n);*
   SWIGTYPE_p_Agedge_t **gv.nextout** *(SWIGTYPE_p_Agnode_t n, SWIGTYPE_p_Agedge_t e);*

Iterate over head nodes reachable from out-edges of a node
   SWIGTYPE_p_Agnode_t **gv.firsthead** *(SWIGTYPE_p_Agnode_t n);*
   SWIGTYPE_p_Agnode_t **gv.nexthead** *(SWIGTYPE_p_Agnode_t n, SWIGTYPE_p_Agnode_t h);*

Iterate over in-edges of a graph
   SWIGTYPE_p_Agedge_t **gv.firstin** *(SWIGTYPE_p_Agraph_t g);*
   SWIGTYPE_p_Agedge_t **gv.nextin** *(SWIGTYPE_p_Agnode_t n, SWIGTYPE_p_Agedge_t e);*

Iterate over in-edges of a node
   SWIGTYPE_p_Agedge_t **gv.firstin** *(SWIGTYPE_p_Agnode_t n);*
   SWIGTYPE_p_Agedge_t **gv.nextin** *(SWIGTYPE_p_Agraph_t g, SWIGTYPE_p_Agedge_t e);*

Iterate over tail nodes reachable from in-edges of a node
   SWIGTYPE_p_Agnode_t **gv.firsttail** *(SWIGTYPE_p_Agnode_t n);*
   SWIGTYPE_p_Agnode_t **gv.nexttail** *(SWIGTYPE_p_Agnode_t n, SWIGTYPE_p_Agnode_t t);*

Iterate over nodes of a graph
   SWIGTYPE_p_Agnode_t **gv.firstnode** *(SWIGTYPE_p_Agraph_t g);*
   SWIGTYPE_p_Agnode_t **gv.nextnode** *(SWIGTYPE_p_Agraph_t g, SWIGTYPE_p_Agnode_t n);*

Iterate over nodes of an edge
   SWIGTYPE_p_Agnode_t **gv.firstnode** *(SWIGTYPE_p_Agedge_t e);*
   SWIGTYPE_p_Agnode_t **gv.nextnode** *(SWIGTYPE_p_Agedge_t e, SWIGTYPE_p_Agnode_t n);*

Iterate over attributes of a graph
   SWIGTYPE_p_Agsym_t **gv.firstattr** *(SWIGTYPE_p_Agraph_t g);*
   SWIGTYPE_p_Agsym_t **gv.nextattr** *(SWIGTYPE_p_Agraph_t g, SWIGTYPE_p_Agsym_t a);*

Iterate over attributes of an edge
   SWIGTYPE_p_Agsym_t **gv.firstattr** *(SWIGTYPE_p_Agedge_t e);*
   SWIGTYPE_p_Agsym_t **gv.nextattr** *(SWIGTYPE_p_Agedge_t e, SWIGTYPE_p_Agsym_t a);*

Iterate over attributes of a node
   SWIGTYPE_p_Agsym_t **gv.firstattr** *(SWIGTYPE_p_Agnode_t n);*
   SWIGTYPE_p_Agsym_t **gv.nextattr** *(SWIGTYPE_p_Agnode_t n, SWIGTYPE_p_Agsym_t a);*

Remove graph objects
   bool **gv.rm** *(SWIGTYPE_p_Agraph_t g);*
   bool **gv.rm** *(SWIGTYPE_p_Agnode_t n);*
   bool **gv.rm** *(SWIGTYPE_p_Agedge_t e);*

Layout

Annotate a graph with layout attributes and values using a specific layout engine
   bool **gv.layout** *(SWIGTYPE_p_Agraph_t g, string engine);*

Render

Render a layout into attributes of the graph
   bool **gv.render** *(SWIGTYPE_p_Agraph_t g);*

Render a layout to stdout
   bool **gv.render** *(SWIGTYPE_p_Agraph_t g, string format);*

Render to an open file
   bool **gv.render** *(SWIGTYPE_p_Agraph_t g, string format, SWIGTYPE_p_FILE fout);*

Render a layout to an unopened file by name
   bool **gv.render** *(SWIGTYPE_p_Agraph_t g, string format, string filename);*

Render to a string result
   string **gv.renderresult** *(SWIGTYPE_p_Agraph_t ing, string format);*
   gv.renderresult *(SWIGTYPE_p_Agraph_t g, string format, string outdata);*

Render to an open channel
   bool **gv.renderchannel** *(SWIGTYPE_p_Agraph_t g, 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** *(SWIGTYPE_p_Agraph_t g, string format);*

Writing graph back to file
   bool **gv.write** *(SWIGTYPE_p_Agraph_t g, string filename);*
   bool **gv.write** *(SWIGTYPE_p_Agraph_t g, SWIGTYPE_p_FILE f);*

Graph transformation tools
   bool **gv.tred** *(SWIGTYPE_p_Agraph_t g);*

KEYWORDS
graph, dot, neato, fdp, circo, twopi, sharp.