NAME

gv_tcl - graph manipulation in tcl

SYNOPSIS

#!/usr/bin/tclsh
package require gv

USAGE

Requires tcl8.3 or later.

INTRODUCTION

gv_tcl is a dynamically loaded extension for **tcl** 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>*
<attr_handle> **gv::findattr** *<graph_handle> <name>*
<attr_handle> **gv::findattr** *<node_handle> <name>*
<attr_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

<boolean_string> **gv::ok** *<graph_handle>*
<boolean_string> **gv::ok** *<node_handle>*
<boolean_string> **gv::ok** *<edge_handle>*
<boolean_string> **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
<attr_handle> **gv::firstattr** *<graph_handle>*
<attr_handle> **gv::nextattr** *<graph_handle> <attr_handle>*

Iterate over attributes of an edge
<attr_handle> **gv::firstattr** *<edge_handle>*
<attr_handle> **gv::nextattr** *<edge_handle> <attr_handle>*

Iterate over attributes of a node
<attr_handle> **gv::firstattr** *<node_handle>*
<attr_handle> **gv::nextattr** *<node_handle> <attr_handle>*

Remove graph objects

<boolean_string> **gv::rm** *<graph_handle>*
<boolean_string> **gv::rm** *<node_handle>*
<boolean_string> **gv::rm** *<edge_handle>*

Layout

Annotate a graph with layout attributes and values using a specific layout engine
<boolean_string> **gv::layout** *<graph_handle> <string> engine*

Render

Render a layout into attributes of the graph

<boolean_string> **gv::render** *<graph_handle>*

Render a layout to stdout

<boolean_string> **gv::render** *<graph_handle> <string> format*

Render to an open file

<boolean_string> **gv::render** *<graph_handle> <string> format <channel> fout*

Render a layout to an unopened file by name

<boolean_string> **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

<boolean_string> **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

<boolean_string> **gv::write** *<graph_handle> <string> filename*
<boolean_string> **gv::write** *<graph_handle> <channel>*

Graph transformation tools

<boolean_string> **gv::tred** *<graph_handle>*

KEYWORDS

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