

**NAME**

`gc` – count graph components

**SYNOPSIS**

`gc [ -necCaDUrsv? ] [ files ]`

**DESCRIPTION**

`gc` is a graph analogue to `wc` in that it prints to standard output the number of nodes, edges, connected components or clusters contained in the input files. It also prints a total count for all graphs if more than one graph is given.

**OPTIONS**

The following options are supported:

- `-n` Count nodes.
- `-e` Count edges.
- `-c` Count connected components.
- `-C` Count clusters. By definition, a cluster is a graph or subgraph whose name begins, case insensitively, with "cluster" or who has the **cluster=true** attribute.
- `-a` Count all. Equivalent to `-encC`
- `-r` Recursively analyze subgraphs.
- `-s` Print no output. Only exit value is important.
- `-D` Only analyze directed graphs.
- `-U` Only analyze undirected graphs.
- `-v` Verbose output.
- `-?` Print usage information.

By default, `gc` returns the number of nodes and edges.

**OPERANDS**

The following operand is supported:

*files* Names of files containing 1 or more graphs in dot format. If no *files* operand is specified, the standard input will be used.

**EXIT STATUS**

The following exit values are returned:

- 0** Successful completion.
- 1** The `-U` or `-E` option was used, and a graph of the wrong type was encountered.

**AUTHOR**

Emden R. Gansner <erg@research.att.com>

**SEE ALSO**

`wc(1)`, `acyclic(1)`, `gvpr(1)`, `gvcolor(1)`, `ccomps(1)`, `sccmap(1)`, `tred(1)`, `libgraph(3)`