

**NAME**

**libgvpr** – library for graph filtering

**SYNOPSIS**

```
#include <graphviz/gvpr.h>

/* If set, gvpr calls exit() on errors */
#define GV_USE_EXIT 1
/* If set, gvpr stores output graphs in gvpropts */
#define GV_USE_OUTGRAPH 2

typedef ssize_t (*gvprwr) (void*, const char *buf, size_t nbyte, void*);

typedef struct {
    Agraph_t** ingraphs; /* NULL-terminated array of input graphs */
    int n_outgraphs; /* if GV_USE_OUTGRAPH set, output graphs */
    Agraph_t** outgraphs;
    gvprwr out; /* write function for stdout */
    gvprwr err; /* write function for stderr */
    int flags;
} gvpropts;

extern int gvpr (int argc, char *argv[], gvpropts* opts);
```

**DESCRIPTION**

The **gvpr** library allows an application to perform general-purpose graph manipulation and filtering based on an awk-like language. (For a more complete description of this language, see `gvpr(1)`.)

The library has a single entry point: the `gvpr()` function. This provides a standard `argc/argv` interface, along with a structure to support in-core graphs, application print functions, along with additional options.

When called, `gvpr()` processes any flags provided in the `argv` array, and compiles the **gvpr** program to be run (provided either via the `-f` flag or as an item in `argv`). It then runs the program on each input graph. If `opt->ingraphs` is non-NULL, this is taken as a NULL-terminated array of in-core graphs to be used as input. Otherwise, the unprocessed elements of `argv` are taken to be the names of files containing graphs to be processed. (If none remain, **gvpr** will read from stdin.)

Normally, **gvpr** writes any output graph to stdout. However, if the flag `GV_USE_OUTGRAPH` is set in `opts->flags`, the output graphs will be stored in an array pointed to be `opts->outgraphs` and the count will be stored in `opts->n_outgraphs`. In this case, the application must call `agclose()` on each output graph when it is done with it.

The application can override the default write functions for stdout and stderr using the `out` and `err` fields in `opts`. When called by **gvpr**, the second argument will point to a buffer of characters to be written, while the third argument provides the number of characters. The function should return the number of bytes actually written.

**RETURN VALUES**

Normally, **gvpr** returns 0 on success and non-zero if an error occurs. Any relevant error message will have been written to stderr or the application's `opts->err` function will have been called. If, however, `GV_USE_EXIT` is set in `opts->flags`, **gvpr** will call `exit(3)` in case of an error.

**SEE ALSO**

`gvpr(1)`, `awk(1)`, `cgraph(3)`

**AUTHORS**

Emden Gansner (erg@research.att.com).