NAME

geom_stats_open, **geom_stats_close**, **geom_stats_resync**, **geom_stats_snapshot_get**, **geom_stats_snapshot_free**, **geom_stats_snapshot_timestamp**, **geom_stats_snapshot_reset**, **geom_stats_snapshot_next**, **gctl_get_handle**, **gctl_ro_param**, **gctl_rw_param**, **gctl_issue**, **gctl_free**, **gctl_dump**, **geom_getxml**, **geom_xml2tree**, **geom_gettree**, **geom_deletetree**, **g_open**, **g_close**, **g_mediasize**, **g_sectorsize**, **g_stripeoffset**, **g_stripesize**, **g_flush**, **g_delete**, **g_device_path**, **g_get_ident**, **g_get_name**, **g_open_by_ident**, **g_providername** **-** userland API library for kernel GEOM subsystem

LIBRARY

Userland API Library for Kernel GEOM subsystem (libgeom, -lgeom)

SYNOPSIS

/* stdio.h is only required for ‘gctl_dump‘ */
#include <stdio.h>
#include <libgeom.h>

Statistics Functions

void
geom_stats_close(*void*);

int
geom_stats_open(*void*);

void
geom_stats_resync(*void*);

*void **
geom_stats_snapshot_get(*void*);

void
geom_stats_snapshot_free(*void *arg*);

void
geom_stats_snapshot_timestamp(*void *arg*, *struct timespec *tp*);

void
geom_stats_snapshot_reset(*void *arg*);

*struct devstat **
geom_stats_snapshot_next(*void *arg*);

Control Functions

*struct gctl_req **
gctl_get_handle(*void*);

void
gctl_ro_param(*struct gctl_req *req*, *const char *name*, *int len*, *const void *value*);

void
gctl_rw_param(*struct gctl_req *req*, *const char *name*, *int len*, *void *value*);

*const char **
gctl_issue(*struct gctl_req *req*);

void
gctl_free(*struct gctl_req *req*);

void
gctl_dump(*struct gctl_req *req*, *FILE *f*);

Utility Functions

*char **
geom_getxml(*void*);

int
geom_xml2tree(*struct gmesh *gmp*, *char *p*);

int
geom_gettree(*struct gmesh *gmp*);

void
geom_deletetree(*struct gmesh *gmp*);

int
g_open(*const char *name*, *int dowrite*);

int
g_close(*int fd*);

off_t
g_mediasize(*int fd*);

ssize_t
g_sectorsize(*int fd*);

ssize_t
g_stripeoffset(*int fd*);

ssize_t
g_stripesize(*int fd*);

int
g_flush(*int fd*);

int
g_delete(*int fd*, *off_t offset*, *off_t length*);

*char **
g_device_path(*const char *devpath*);

int
g_get_ident(*int fd*, *char *ident*, *size_t size*);

int
g_get_name(*const char *ident*, *char *name*, *size_t size*);

int
g_open_by_ident(*const char *ident*, *int dowrite*, *char *name*, *size_t size*);

*char **
g_providername(*int fd*);

DESCRIPTION

The **geom** library contains the official and publicized API for interacting with the GEOM subsystem in the kernel.

Statistics Functions

GEOM collects statistics data for all consumers and providers, but does not perform any normalization or presentation on the raw data, this is left as an exercise for user-land presentation utilities.

The **geom_stats_open**() and **geom_stats_close**() functions open and close the necessary pathways to access the raw statistics information in the kernel. These functions are likely to open one or more files and cache the file descriptors locally. The **geom_stats_open**() function returns zero on success, and sets *errno* if not.

The **geom_stats_resync**() function will check if more statistics collection points have been added in the kernel since **geom_stats_open**() or the previous call to **geom_stats_resync**().

The **geom_stats_snapshot_get**() function will acquire a snapshot of the raw data from the kernel, and while a reasonable effort is made to make this snapshot as atomic and consistent as possible, no guarantee is given that it will actually be so. The snapshot must be freed again using the **geom_stats_snapshot_free**() function. The **geom_stats_snapshot_get**() function returns NULL on failure.

The **geom_stats_snapshot_timestamp**() function provides access to the timestamp acquired in the snapshot.

The **geom_stats_snapshot_reset**() and **geom_stats_snapshot_next**() functions provide an iterator over the statistics slots in the snapshot. The **geom_stats_snapshot_reset**() function forces the internal pointer in the snapshot back to before the first item. The **geom_stats_snapshot_next**() function returns the next item, and NULL if there are no more items in the snapshot.

Control Functions

The **gctl_***() functions are used to send requests to GEOM classes. In order for a GEOM class to actually be able to receive these requests, it must have defined a "ctlreq" method.

A *struct gctl_req **, obtained with **gctl_get_handle**(), can hold any number of parameters, which must be added to it with **gctl_ro_param**() (for read-only parameters) or **gctl_rw_param**() (for read/write parameters).

Both **gctl_ro_param**() and **gctl_rw_param**() take a string *name*, which is used to identify the parameter, and a *value*, which contains, in the read-only case, the data to be passed to the GEOM class, or, in the read/write case, a pointer to preallocated memory that the GEOM class should fill with the desired data. If *len* is negative, it is assumed that *value* is an ASCII string and the actual length is taken from the string length of *value*; otherwise it must hold the size of *value*.

A parameter with a *name* containing the string "class" is mandatory for each request, and the corresponding *value* must hold the name of the GEOM class where the request should be sent to.

Also mandatory for each request is a parameter with a *name* called "verb", and the corresponding *value* needs to hold the command string that the GEOM class should react upon.

Once all desired parameters are filled in, the request must be sent to the GEOM subsystem with **gctl_issue**(), which returns NULL on success, or a string containing the error message on failure.

After the request is finished, the allocated memory should be released with **gctl_free**().

The **gctl_dump**() function can be used to format the contents of *req* to the open file handle pointed to by *f*, for debugging purposes.

Error handling for the control functions is postponed until the call to **gctl_issue**(), which returns NULL on success, or an error message corresponding to the first error which happened.

Utility Functions

The **geom_getxml**() function is a wrapper around sysctl(3) that fetches the *kern.geom.confxml* OID, and returns it's value. The allocated memory should be released with free(3) after use.

The **geom_xml2tree**() function parses the XML representation of a GEOM topology passed as *p*, allocates the needed data structures to access this information and fills in the passed *gmp* data structure. Memory allocated during this transformation should be released using **geom_deletetree**() after use.

The **geom_gettree**() function is a wrapper around the **geom_getxml**() and **geom_xml2tree**() functions. Memory allocated during this operation should be released using **geom_deletetree**() after use.

The **geom_deletetree**() function releases memory allocated for storing the data-structures referenced by *gmp*.

The **g_***() functions are used to communicate with GEOM providers.

The **g_open**() function opens the given provider and returns file descriptor number, which can be used with other functions. The *dowrite* argument indicates if operations that modify the provider (like **g_flush**() or **g_delete**()) are going to be called.

The **g_close**() function closes the provider.

The **g_mediasize**() function returns size of the given provider.

The **g_sectorsize**() function returns sector size of the given provider.

The **g_stripeoffset**() function returns stripe offset of the given provider.

The **g_stripesize**() function returns stripe size of the given provider.

The **g_flush**() function sends BIO_FLUSH request to flush write cache of the provider.

The **g_delete**() function tells the provider that the given data range is no longer used.

The **g_device_path**() function returns the full path to a provider given a partial or full path to the device node. NULL is returned if the device cannot be found or is not a valid geom provider.

The **g_get_ident**() function returns provider's fixed and unique identifier. The *ident* argument should be at least DISK_IDENT_SIZE big.

The **g_get_name**() function returns name of the provider, which identifier is equal to the *ident* string.

The **g_open_by_ident**() function opens provider using its identification, unlike **g_open**() which uses the provider's name. The function will store the provider's name in the *name* parameter if it is not NULL.

The **g_providername**() function returns the provider name of an open file descriptor. NULL is returned the file descriptor does not point to a valid geom provider.

All functions except **g_providername**() and **g_device_path**() return a value greater than or equal to *0* on success or *-1* on failure.

EXAMPLES

Create a request that is to be sent to the CCD class, and tell it to destroy a specific geom:

```
H = gctl_get_handle();
gctl_ro_param(H, "verb", -1, "destroy geom");
gctl_ro_param(H, "class", -1, "CCD");
sprintf(buf, "ccd%d", ccd);
gctl_ro_param(H, "geom", -1, buf);
errstr = gctl_issue(H);
if (errstr != NULL)
    err(1, "could not destroy ccd: %s", errstr);
gctl_free(H);
```

HISTORY

The **geom** library appeared in FreeBSD 5.1.

AUTHORS

Poul-Henning Kamp <*phk@FreeBSD.org*>
Lukas Ertl <*le@FreeBSD.org*>

Pawel Jakub Dawidek <*pjd@FreeBSD.org*>