NAME

disk - kernel disk storage API

SYNOPSIS

#include <geom/geom_disk.h>

*struct disk **
disk_alloc(*void*);

void
disk_create(*struct disk *disk, int version*);

void
disk_gone(*struct disk *disk*);

void
disk_destroy(*struct disk *disk*);

int
disk_resize(*struct disk *disk, int flags*);

void
disk_add_alias(*struct disk *disk, const char *alias*);

DESCRIPTION

The disk storage API permits kernel device drivers providing access to disk-like storage devices to advertise the device to other kernel components, including GEOM(4) and devfs(5).

Each disk device is described by a *struct disk* structure, which contains a variety of parameters for the disk device, function pointers for various methods that may be performed on the device, as well as private data storage for the device driver. In addition, some fields are reserved for use by GEOM in managing access to the device and its statistics.

GEOM has the ownership of *struct disk*, and drivers must allocate storage for it with the **disk_alloc**() function, fill in the fields and call **disk_create**() when the device is ready to service requests. **disk_add_alias**() adds an alias for the disk and must be called before **disk_create**(), but may be called multiple times. For each alias added, a device node will be created with make_dev_alias(9) in the same way primary device nodes are created with make_dev(9) for *d_name* and *d_unit*. Care should be taken to ensure that only one driver creates aliases for any given name. **disk_resize**() can be called by the driver after modifying *d_mediasize* to notify GEOM about the disk capacity change. The *flags* field

should be set to either M_WAITOK, or M_NOWAIT. **disk_gone**() orphans all of the providers associated with the drive, setting an error condition of ENXIO in each one. In addition, it prevents a re-taste on last close for writing if an error condition has been set in the provider. After calling **disk_destroy**(), the device driver is not allowed to access the contents of *struct disk* anymore.

The **disk_create**() function takes a second parameter, *version*, which must always be passed DISK_VERSION. If GEOM detects that the driver is compiled against an unsupported version, it will ignore the device and print a warning on the console.

Descriptive Fields

The following fields identify the disk device described by the structure instance, and must be filled in prior to submitting the structure to **disk_create**() and may not be subsequently changed:

u_int d_flags

Optional flags indicating to the storage framework what optional features or descriptions the storage device driver supports. Currently supported flags are DISKFLAG_OPEN (maintained by storage framework), DISKFLAG_CANDELETE (maintained by device driver), and DISKFLAG_CANFLUSHCACHE (maintained by device driver).

*const char * d_name*

Holds the name of the storage device class, e.g., "ahd". This value typically uniquely identifies a particular driver device, and must not conflict with devices serviced by other device drivers.

u_int d_unit

Holds the instance of the storage device class, e.g., "4". This namespace is managed by the device driver, and assignment of unit numbers might be a property of probe order, or in some cases topology. Together, the *d_name* and *d_unit* values will uniquely identify a disk storage device.

Disk Device Methods

The following fields identify various disk device methods, if implemented:

*disk_open_t * d_open*

Optional: invoked when the disk device is opened. If no method is provided, open will always succeed.

*disk_close_t * d_close*

Optional: invoked when the disk device is closed. Although an error code may be returned, the call should always terminate any state setup by the corresponding open method call.

*disk_strategy_t * d_strategy*

Mandatory: invoked when a new *struct bio* is to be initiated on the disk device.

*disk_ioctl_t * d_ioctl*

Optional: invoked when an I/O control operation is initiated on the disk device. Please note that for security reasons these operations should not be able to affect other devices than the one on which they are performed.

*dumper_t * d_dump*

Optional: if configured with dumpon(8), this function is invoked from a very restricted system state after a kernel panic to record a copy of the system RAM to the disk.

*disk_getattr_t * d_getattr*

Optional: if this method is provided, it gives the disk driver the opportunity to override the default GEOM response to BIO_GETATTR requests. This function should return -1 if the attribute is not handled, 0 if the attribute is handled, or an errno to be passed to **g_io_deliver**().

*disk_gone_t * d_gone*

Optional: if this method is provided, it will be called after **disk_gone**() is called, once GEOM has finished its cleanup process. Once this callback is called, it is safe for the disk driver to free all of its resources, as it will not be receiving further calls from GEOM.

Mandatory Media Properties

The following fields identify the size and granularity of the disk device. These fields must stay stable from return of the drivers open method until the close method is called, but it is perfectly legal to modify them in the open method before returning.

u_int d_sectorsize

The sector size of the disk device in bytes.

off_t d_mediasize

The size of the disk device in bytes.

u_int d_maxsize

The maximum supported size in bytes of an I/O request. Requests larger than this size will be chopped up by GEOM.

Optional Media Properties

These optional fields can provide extra information about the disk device. Do not initialize these fields if the field/concept does not apply. These fields must stay stable from return of the drivers open method

until the close method is called, but it is perfectly legal to modify them in the open method before returning.

u_int d_fwsectors, u_int d_fwheads
: The number of sectors and heads advertised on the disk device by the firmware or BIOS. These values are almost universally bogus, but on some architectures necessary for the correct calculation of disk partitioning.

u_int d_stripeoffset, u_int d_stripesize
: These two fields can be used to describe the width and location of natural performance boundaries for most disk technologies. Please see *src/sys/geom/notes* for details.

char d_ident[DISK_IDENT_SIZE]
: This field can and should be used to store disk's serial number if the d_getattr method described above isn't implemented, or if it does not support the GEOM::ident attribute.

char d_descr[DISK_IDENT_SIZE]
: This field can be used to store the disk vendor and product description.

uint16_t d_hba_vendor
: This field can be used to store the PCI vendor ID for the HBA connected to the disk.

uint16_t d_hba_device
: This field can be used to store the PCI device ID for the HBA connected to the disk.

uint16_t d_hba_subvendor
: This field can be used to store the PCI subvendor ID for the HBA connected to the disk.

uint16_t d_hba_subdevice
: This field can be used to store the PCI subdevice ID for the HBA connected to the disk.

Driver Private Data

This field may be used by the device driver to store a pointer to private data to implement the disk service.

*void * d_drv1*
: Private data pointer. Typically used to store a pointer to the drivers *softc* structure for this disk device.

SEE ALSO

GEOM(4), devfs(5)

HISTORY

The **kernel disk storage API** first appeared in FreeBSD 4.9.

AUTHORS

This manual page was written by Robert Watson.

BUGS

Disk aliases are not a general purpose aliasing mechanism, but are intended only to ease the transition from one name to another. They can be used to ensure that nvd0 and nda0 are the same thing. They cannot be used to implement the diskX concept from macOS.