NAME

devinfo, **devinfo_init**, **devinfo_free**, **devinfo_handle_to_device**, **devinfo_handle_to_resource**, **devinfo_handle_to_rman**, **devinfo_foreach_device_child**, **devinfo_foreach_device_resource**, **devinfo_foreach_rman_resource**, **devinfo_foreach_rman** - device and resource information utility library

LIBRARY

Device and Resource Information Utility Library (libdevinfo, -ldevinfo)

SYNOPSIS

#include <devinfo.h>

int
devinfo_init(*void*);

void
devinfo_free(*void*);

*struct devinfo_dev **
devinfo_handle_to_device(*devinfo_handle_t handle*);

*struct devinfo_res **
devinfo_handle_to_resource(*devinfo_handle_t handle*);

*struct devinfo_rman **
devinfo_handle_to_rman(*devinfo_handle_t handle*);

int
devinfo_foreach_device_child(*struct devinfo_dev *parent,*
*int (*fn)(struct devinfo_dev *child, void *arg), void *arg*);

int
devinfo_foreach_device_resource(*struct devinfo_dev *dev, int (*fn)(struct devinfo_dev *dev,*
*struct devinfo_res *res, void *arg), void *arg*);

int
devinfo_foreach_rman_resource(*struct devinfo_rman *rman,*
*int (*fn)(struct devinfo_res *res, void *arg), void *arg*);

int
devinfo_foreach_rman(*int (*fn)(struct devinfo_rman *rman, void *arg), void *arg*);

DESCRIPTION

The **devinfo** library provides access to the kernel's internal device hierarchy and to the I/O resource manager. The library uses a sysctl(3) interface to obtain a snapshot of the kernel's state, which is then made available to the application.

Due to the fact that the information may be logically arranged in a number of different fashions, the library does not attempt to impose any structure on the data.

Device, resource, and resource manager information is returned in data structures defined in *<devinfo.h>*:

```
struct devinfo_dev {
    devinfo_handle_t    dd_handle;          /* device handle */
    devinfo_handle_t    dd_parent;          /* parent handle */
    char                *dd_name;           /* name of device */
    char                *dd_desc;           /* device description */
    char                *dd_drivername;     /* name of attached driver */
    char                *dd_pnpinfo;        /* pnp info from parent bus */
    char                *dd_location;       /* Where bus thinks dev at */
    uint32_t                    dd_devflags;        /* API flags */
    uint16_t                    dd_flags; /* internal dev flags */
    device_state_t      dd_state; /* attachment state of dev */
};

struct devinfo_rman {
    devinfo_handle_t    dm_handle;          /* resource manager handle */
    rman_res_t                  dm_start; /* resource start */
    rman_res_t                  dm_size;  /* resource size */
    char                *dm_desc;           /* resource description */
};

struct devinfo_res {
    devinfo_handle_t    dr_handle;          /* resource handle */
    devinfo_handle_t    dr_rman; /* resource manager handle */
    devinfo_handle_t    dr_device;          /* owning device */
    rman_res_t                  dr_start;   /* region start */
    rman_res_t                  dr_size;    /* region size */
};
```

The *devinfo_handle_t* values can be used to look up the correspondingly referenced structures.

devinfo_init() takes a snapshot of the kernel's internal device and resource state. It returns nonzero if after a number of retries a consistent snapshot cannot be obtained. **devinfo_init**() must be called before any other functions can be used.

devinfo_free() releases the memory associated with the snapshot. Any pointers returned by other functions are invalidated by this, and **devinfo_init**() must be called again before using any other functions.

devinfo_handle_to_device(), **devinfo_handle_to_resource**() and **devinfo_handle_to_rman**() return pointers to *devinfo_dev*, *devinfo_res* and *devinfo_rman* structures respectively based on the *devinfo_handle_t* passed to them. These functions can be used to traverse the tree from any node to any other node. If **devinfo_handle_to_device**() is passed the constant DEVINFO_ROOT_DEVICE it will return the handle to the root of the device tree.

devinfo_foreach_device_child() invokes its callback argument *fn* on every device which is an immediate child of *device*. The *fn* function is also passed *arg*, allowing state to be passed to the callback function. If *fn* returns a nonzero error value the traversal is halted, and **devinfo_foreach_device_child**() returns the error value to its caller.

devinfo_foreach_device_resource() invokes its callback argument *fn* on every resource which is owned by *device*. The *fn* function is also passed *device* and *arg*, allowing state to be passed to the callback function. If *fn* returns a nonzero error value the traversal is halted, and **devinfo_foreach_device_resource**() returns the error value to its caller.

devinfo_foreach_rman_resource() invokes its callback argument *fn* on every resource within the resource manager *rman*. The *fn* function is also passed *arg*, allowing state to be passed to the callback function. If *fn* returns a nonzero error value the traversal is halted, and **devinfo_foreach_rman_resource**() returns the error value to its caller.

devinfo_foreach_rman() invokes its callback argument *fn* on every resource manager. The *fn* function is also passed *arg*, allowing state to be passed to the callback function. If *fn* returns a nonzero error value the traversal is halted, and **devinfo_foreach_rman**() returns the error value to its caller.

SEE ALSO

devstat(3)

HISTORY

The **devinfo** library first appeared in FreeBSD 5.0.

AUTHORS

Michael Smith <*msmith@FreeBSD.org*>

BUGS

This is the first implementation of the library, and the interface is still subject to refinement.

The interface does not report device classes or drivers, making it hard to sort by class or driver.