NAME

make_dev, **make_dev_cred**, **make_dev_credf**, **make_dev_p**, **make_dev_s**, **make_dev_alias**, **make_dev_alias_p**, **destroy_dev**, **destroy_dev_sched**, **destroy_dev_sched_cb**, **destroy_dev_drain**, **dev_depends** - manage *cdev*'s and DEVFS registration for devices

SYNOPSIS

#include <sys/param.h>
#include <sys/conf.h>

void
make_dev_args_init(*struct make_dev_args *args*);

int
make_dev_s(*struct make_dev_args *args, struct cdev **cdev, const char *fmt, ...*);

int
make_dev_alias_p(*int flags, struct cdev **cdev, struct cdev *pdev, const char *fmt, ...*);

void
destroy_dev(*struct cdev *dev*);

void
destroy_dev_sched(*struct cdev *dev*);

void
destroy_dev_sched_cb(*struct cdev *dev, void (*cb)(void *), void *arg*);

void
destroy_dev_drain(*struct cdevsw *csw*);

void
dev_depends(*struct cdev *pdev, struct cdev *cdev*);

LEGACY INTERFACES
*struct cdev **
make_dev(*struct cdevsw *cdevsw, int unit, uid_t uid, gid_t gid, int perms, const char *fmt, ...*);

*struct cdev **
make_dev_cred(*struct cdevsw *cdevsw, int unit, struct ucred *cr, uid_t uid, gid_t gid, int perms, const char *fmt, ...*);

*struct cdev **
make_dev_credf(*int flags, struct cdevsw *cdevsw, int unit, struct ucred *cr, uid_t uid, gid_t gid, int perms, const char *fmt, ...*);

int
make_dev_p(*int flags, struct cdev **cdev, struct cdevsw *devsw, struct ucred *cr, uid_t uid, gid_t gid, int mode, const char *fmt, ...*);

*struct cdev **
make_dev_alias(*struct cdev *pdev, const char *fmt, ...*);

DESCRIPTION

The **make_dev_s**() function creates a *cdev* structure for a new device, which is returned into the *cdev* argument. It also notifies devfs(5) of the presence of the new device, that causes corresponding nodes to be created. Besides this, a devctl(4) notification is sent. The function takes the structure *struct make_dev_args args*, which specifies the parameters for the device creation:

```
struct make_dev_args {
        size_t          mda_size;
        int             mda_flags;
        struct cdevsw   *mda_devsw;
        struct ucred    *mda_cr;
        uid_t           mda_uid;
        gid_t           mda_gid;
        int             mda_mode;
        int             mda_unit;
        void            *mda_si_drv1;
        void            *mda_si_drv2;
};
```

Before use and filling with the desired values, the structure must be initialized by the **make_dev_args_init**() function, which ensures that future kernel interface expansion does not affect driver source code or binary interface.

The created device will be owned by *args.mda_uid*, with the group ownership as *args.mda_gid*. The name is the expansion of *fmt* and following arguments as printf(9) would print it. The name determines its path under */dev* or other devfs(5) mount point and may contain slash '/' characters to denote subdirectories. The permissions of the file specified in *args.mda_mode* are defined in <*sys/stat.h*>:

```
#define S_IRWXU 0000700    /* RWX mask for owner */
#define S_IRUSR 0000400    /* R for owner */
```

```
#define S_IWUSR 0000200    /* W for owner */
#define S_IXUSR 0000100    /* X for owner */

#define S_IRWXG 0000070    /* RWX mask for group */
#define S_IRGRP 0000040    /* R for group */
#define S_IWGRP 0000020    /* W for group */
#define S_IXGRP 0000010    /* X for group */

#define S_IRWXO 0000007    /* RWX mask for other */
#define S_IROTH 0000004    /* R for other */
#define S_IWOTH 0000002    /* W for other */
#define S_IXOTH 0000001    /* X for other */

#define S_ISUID 0004000    /* set user id on execution */
#define S_ISGID 0002000    /* set group id on execution */
#define S_ISVTX 0001000    /* sticky bit */
#ifndef _POSIX_SOURCE
#define S_ISTXT 0001000
#endif
```

The *args.mda_cr* argument specifies credentials that will be stored in the *si_cred* member of the initialized *struct cdev*.

The *args.mda_flags* argument alters the operation of **make_dev_s.**() The following values are currently accepted:

MAKEDEV_REF	reference the created device
MAKEDEV_NOWAIT	do not sleep, the call may fail
MAKEDEV_WAITOK	allow the function to sleep to satisfy malloc
MAKEDEV_ETERNAL	created device will be never destroyed
MAKEDEV_CHECKNAME	return an error if the device name is invalid or already exists

Only MAKEDEV_NOWAIT, MAKEDEV_WAITOK and MAKEDEV_CHECKNAME values are accepted for the **make_dev_alias_p**() function.

The MAKEDEV_WAITOK flag is assumed if none of MAKEDEV_WAITOK, MAKEDEV_NOWAIT is specified.

The dev_clone(9) event handler shall specify MAKEDEV_REF flag when creating a device in response to lookup, to avoid race where the device created is destroyed immediately after devfs_lookup(9) drops

his reference to cdev.

The MAKEDEV_ETERNAL flag allows the kernel to not acquire some locks when translating system calls into the cdevsw methods calls. It is responsibility of the driver author to make sure that **destroy_dev**() is never called on the returned cdev. For the convenience, use the MAKEDEV_ETERNAL_KLD flag for the code that can be compiled into kernel or loaded (and unloaded) as loadable module.

A panic will occur if the MAKEDEV_CHECKNAME flag is not specified and the device name is invalid or already exists.

The **make_dev_p**() use of the form

```
struct cdev *dev;
int res;
res = make_dev_p(flags, &dev, cdevsw, cred, uid, gid, perms, name);
```

is equivalent to the code

```
struct cdev *dev;
struct make_dev_args args;
int res;

make_dev_args_init(&args);
args.mda_flags = flags;
args.mda_devsw = cdevsw;
args.mda_cred = cred;
args.mda_uid = uid;
args.mda_gid = gid;
args.mda_mode = perms;
res = make_dev_s(&args, &dev, name);
```

Similarly, the **make_dev_credf**() function call is equivalent to

```
(void) make_dev_s(&args, &dev, name);
```

In other words, **make_dev_credf**() does not allow the caller to obtain the return value, and in kernels compiled with the *INVARIANTS* options, the function asserts that the device creation succeeded.

The **make_dev_cred**() function is equivalent to the call

```
make_dev_credf(0, cdevsw, unit, cr, uid, gid, perms, fmt, ...);
```

The **make_dev**() function call is the same as

```
make_dev_credf(0, cdevsw, unit, NULL, uid, gid, perms, fmt, ...);
```

The **make_dev_alias_p**() function takes the returned *cdev* from **make_dev**() and makes another (aliased) name for this device. It is an error to call **make_dev_alias_p**() prior to calling **make_dev**().

The **make_dev_alias**() function is similar to **make_dev_alias_p**() but it returns the resulting aliasing **cdev* and may not return an error.

The *cdev* returned by **make_dev_s**() and **make_dev_alias_p**() has two fields, *si_drv1* and *si_drv2*, that are available to store state. Both fields are of type *void **, and can be initialized simultaneously with the *cdev* allocation by filling *args.mda_si_drv1* and *args.mda_si_drv2* members of the **make_dev_s**() argument structure, or filled after the *cdev* is allocated, if using legacy interfaces. In the latter case, the driver should handle the race of accessing uninitialized *si_drv1* and *si_drv2* itself. These are designed to replace the *unit* argument to **make_dev**(), which can be obtained with **dev2unit**().

The **destroy_dev**() function takes the returned *cdev* from **make_dev**() and destroys the registration for that device. The notification is sent to devctl(4) about the destruction event. Do not call **destroy_dev**() on devices that were created with **make_dev_alias**().

The **dev_depends**() function establishes a parent-child relationship between two devices. The net effect is that a **destroy_dev**() of the parent device will also result in the destruction of the child device(s), if any exist. A device may simultaneously be a parent and a child, so it is possible to build a complete hierarchy.

The **destroy_dev_sched_cb**() function schedules execution of the **destroy_dev**() for the specified *cdev* in the safe context. After **destroy_dev**() is finished, and if the supplied *cb* is not NULL, the callback *cb* is called, with argument *arg*. The **destroy_dev_sched**() function is the same as

```
destroy_dev_sched_cb(cdev, NULL, NULL);
```

The **d_close**() driver method cannot call **destroy_dev**() directly. Doing so causes deadlock when **destroy_dev**() waits for all threads to leave the driver methods. Also, because **destroy_dev**() sleeps, no non-sleepable locks may be held over the call. The **destroy_dev_sched**() family of functions overcome these issues.

The device driver may call the **destroy_dev_drain**() function to wait until all devices that have supplied *csw* as cdevsw, are destroyed. This is useful when driver knows that **destroy_dev_sched**() is called for all instantiated devices, but need to postpone module unload until **destroy_dev**() is actually finished for

all of them.

RETURN VALUES

If successful, **make_dev_s**() and **make_dev_p**() will return 0, otherwise they will return an error. If successful, **make_dev_credf**() will return a valid *cdev* pointer, otherwise it will return NULL.

ERRORS

The **make_dev_s**(), **make_dev_p**() and **make_dev_alias_p**() calls will fail and the device will be not registered if:

[ENOMEM] The MAKEDEV_NOWAIT flag was specified and a memory allocation request could not be satisfied.

[ENAMETOOLONG] The MAKEDEV_CHECKNAME flag was specified and the provided device name is longer than SPECNAMELEN.

[EINVAL] The MAKEDEV_CHECKNAME flag was specified and the provided device name is empty, contains a "." or ".." path component or ends with '/'.

[EINVAL] The MAKEDEV_CHECKNAME flag was specified and the provided device name contains invalid characters.

[EEXIST] The MAKEDEV_CHECKNAME flag was specified and the provided device name already exists.

SEE ALSO

devctl(4), devfs(5), dev_clone(9)

HISTORY

The **make_dev**() and **destroy_dev**() functions first appeared in FreeBSD 4.0. The function **make_dev_alias**() first appeared in FreeBSD 4.1. The function **dev_depends**() first appeared in FreeBSD 5.0. The functions **make_dev_credf**(), **destroy_dev_sched**(), **destroy_dev_sched_cb**() first appeared in FreeBSD 7.0. The function **make_dev_p**() first appeared in FreeBSD 8.2. The function **make_dev_s**() first appeared in FreeBSD 11.0.