

**NAME**

**libcuse** - Userland character device library

**LIBRARY**

Userland Character Device Library (libcuse, -lcuse)

**SYNOPSIS**

To load the required kernel module at boot time, place the following line in loader.conf(5):

```
cuse_load="YES"
```

```
#include <cuse.h>
```

**DESCRIPTION**

The **libcuse** library contains functions to create a character device in userspace. The **libcuse** library is thread safe.

**LIBRARY INITIALISATION / DEINITIALISATION**

*int* **cuse\_init**(*void*) This function initialises **libcuse**. Must be called at the beginning of the program. This function returns 0 on success or a negative value on failure. See CUSE\_ERR\_XXX for known error codes. If the cuse kernel module is not loaded, CUSE\_ERR\_NOT\_LOADED is returned.

*int* **cuse\_uninit**(*void*) Deinitialise **libcuse**. Can be called at the end of the application. This function returns 0 on success or a negative value on failure. See CUSE\_ERR\_XXX for known error codes.

**UNIT MANAGEMENT**

*int* **cuse\_alloc\_unit\_number**(*int* \*) This function stores a uniq system unit number at the pointed integer location. This function returns 0 on success or a negative value on failure. See CUSE\_ERR\_XXX for known error codes.

*int* **cuse\_alloc\_unit\_number\_by\_id**(*int* \*, *int* *id*) This function stores a unique system unit number at the pointed integer location. The returned unit number is uniq within the given ID. Valid ID values are defined by the cuse include file. See the **CUSE\_ID\_XXX()** macros for more information. This function returns 0 on success or a negative value on failure. See CUSE\_ERR\_XXX for known error codes.

*int* **cuse\_free\_unit\_number**(*int*) This function frees the given allocated system unit number. This function returns 0 on success or a negative value on failure. See CUSE\_ERR\_XXX for known error codes.

*int* **cuse\_free\_unit\_number\_by\_id**(*int* *unit*, *int* *id*) This function frees the given allocated system unit

number belonging to the given ID. If both the unit and id argument is -1, all allocated units will be freed. This function returns 0 on success or a negative value on failure. See CUSE\_ERR\_XXX for known error codes.

## LIBRARY USAGE

*void \* **cuse\_vmalloc**(*unsigned size*)* This function allocates *size* bytes of memory. Only memory allocated by this function can be memory mapped by `mmap(2)`. This function returns a valid data pointer on success or NULL on failure. The returned pointer is always aligned to the system page size. The number and size of allocations is limited by the `mmap(2)` offset having to fit into a 32-bit variable typically for 32-bit applications.

*int **cuse\_is\_vmalloc\_addr**(*void \**)* This function returns non-zero if the passed pointer points to a valid and non-freed allocation, as returned by `cuse_vmalloc()`. Else this function returns zero.

*void **cuse\_vmfree**(*void \**)* This function frees memory allocated by `cuse_vmalloc()`. This function is NULL safe.

*unsigned long **cuse\_vmoffset**(*void \**)* This function returns the `mmap` offset the client must use to access the allocated memory. The passed pointer must be aligned to the system page size.

*struct cuse\_dev \* **cuse\_dev\_create**(*const struct cuse\_methods \*mtod, void \*priv0, void \*priv1, uid\_t, gid\_t, int permission, const char \*fmt, ...*)* This function creates a new character device according to the given parameters. This function returns a valid `cuse_dev` structure pointer on success or NULL on failure. The device name can only contain a-z, A-Z, 0-9, dot, / and underscore characters.

*void **cuse\_dev\_destroy**(*struct cuse\_dev \**)* This functions destroys a previously created character device.

*void \* **cuse\_dev\_get\_priv0**(*struct cuse\_dev \**), void \* **cuse\_dev\_get\_priv1**(*struct cuse\_dev \**), void **cuse\_dev\_set\_priv0**(*struct cuse\_dev \*, void \**), void **cuse\_dev\_set\_priv1**(*struct cuse\_dev \*, void \**)*  
These functions are used to set and get the private data of the given cuse device.

*int **cuse\_wait\_and\_process**(*void*)* This function will block and do event processing. If parallel I/O is required multiple threads must be created looping on this function. This function returns 0 on success or a negative value on failure. See CUSE\_ERR\_XXX for known error codes.

*void \* **cuse\_dev\_get\_per\_file\_handle**(*struct cuse\_dev \**), void **cuse\_dev\_set\_per\_file\_handle**(*struct cuse\_dev \*, void \**)* These functions are used to set and get the per-file-open specific handle and should only be used inside the cuse file operation callbacks.

*void **cuse\_set\_local**(*int*)* This function instructs `cuse_copy_out()` and `cuse_copy_in()` that the user

pointer is local, if the argument passed to it is non-zero. Else the user pointer is assumed to be at the peer application. This function should only be used inside the cuse file operation callbacks. The value is reset to zero when the given file operation returns, and does not affect any other file operation callbacks.

*int* **cuse\_get\_local**(*void*) Returns the current local state. See **cuse\_set\_local**().

*int* **cuse\_copy\_out**(*const void \*src, void \*peer\_dst, int len*), *int* **cuse\_copy\_in**(*const void \*peer\_src, void \*dst, int len*) These functions are used to transfer data between the local application and the peer application. These functions must be used when operating on the data pointers passed to the **cm\_read**(), **cm\_write**(), and **cm\_ioctl**() callback functions. These functions return 0 on success or a negative value on failure. See CUSE\_ERR\_XXX for known error codes.

*int* **cuse\_got\_peer\_signal**(*void*) This function is used to check if a signal has been delivered to the peer application and should only be used inside the cuse file operation callbacks. This function returns 0 if a signal has been delivered to the caller. Else it returns a negative value. See CUSE\_ERR\_XXX for known error codes.

*struct cuse\_dev \** **cuse\_dev\_get\_current**(*int \*pcmd*) This function is used to get the current cuse device pointer and the currently executing command, by CUSE\_CMD\_XXX value. The *pcmd* argument is allowed to be NULL. This function should only be used inside the cuse file operation callbacks. On success a valid cuse device pointer is returned. On failure NULL is returned.

*void* **cuse\_poll\_wakeup**(*void*) This function will wake up any file pollers.

## LIBRARY LIMITATIONS

Transfer lengths for **read**(), **write**(), **cuse\_copy\_in**(), and **cuse\_copy\_out**() should not exceed what can fit into a 32-bit signed integer and is defined by the **CUSE\_LENGTH\_MAX**() macro. Transfer lengths for **ioctl**s should not exceed what is defined by the **CUSE\_BUFFER\_MAX**() macro.

## LIBRARY CALLBACK METHODS

In general fflags are defined by CUSE\_FFLAG\_XXX and errors are defined by CUSE\_ERR\_XXX.

```
enum {
    CUSE_ERR_NONE
    CUSE_ERR_BUSY
    CUSE_ERR_WOULDBLOCK
    CUSE_ERR_INVALID
    CUSE_ERR_NO_MEMORY
    CUSE_ERR_FAULT
}
```

```
CUSE_ERR_SIGNAL
CUSE_ERR_OTHER
CUSE_ERR_NOT_LOADED
CUSE_ERR_NO_DEVICE
```

```
CUSE_POLL_NONE
CUSE_POLL_READ
CUSE_POLL_WRITE
CUSE_POLL_ERROR
```

```
CUSE_FFLAG_NONE
CUSE_FFLAG_READ
CUSE_FFLAG_WRITE
CUSE_FFLAG_NONBLOCK
CUSE_FFLAG_COMPAT32
```

```
CUSE_CMD_NONE
CUSE_CMD_OPEN
CUSE_CMD_CLOSE
CUSE_CMD_READ
CUSE_CMD_WRITE
CUSE_CMD_IOCTL
CUSE_CMD_POLL
CUSE_CMD_SIGNAL
CUSE_CMD_SYNC
CUSE_CMD_MAX
```

```
};
```

*int* **cuse\_open\_t**(*struct cuse\_dev \**, *int fflags*) This function returns a CUSE\_ERR\_XXX value.

*int* **cuse\_close\_t**(*struct cuse\_dev \**, *int fflags*) This function returns a CUSE\_ERR\_XXX value.

*int* **cuse\_read\_t**(*struct cuse\_dev \**, *int fflags*, *void \*peer\_ptr*, *int len*) This function returns a CUSE\_ERR\_XXX value in case of failure or the actually transferred length in case of success. **cuse\_copy\_in()** and **cuse\_copy\_out()** must be used to transfer data to and from the *peer\_ptr*.

*int* **cuse\_write\_t**(*struct cuse\_dev \**, *int fflags*, *const void \*peer\_ptr*, *int len*) This function returns a CUSE\_ERR\_XXX value in case of failure or the actually transferred length in case of success. **cuse\_copy\_in()** and **cuse\_copy\_out()** must be used to transfer data to and from the *peer\_ptr*.

*int* **cuse\_ioctl\_t**(*struct cuse\_dev \**, *int fflags*, *unsigned long cmd*, *void \*peer\_data*) This function returns a CUSE\_ERR\_XXX value in case of failure or zero in case of success. **cuse\_copy\_in()** and **cuse\_copy\_out()** must be used to transfer data to and from the *peer\_data*.

*int* **cuse\_poll\_t**(*struct cuse\_dev \**, *int fflags*, *int events*) This function returns a mask of CUSE\_POLL\_XXX values in case of failure and success. The events argument is also a mask of CUSE\_POLL\_XXX values.

```
struct cuse_methods {
    cuse_open_t *cm_open;
    cuse_close_t *cm_close;
    cuse_read_t *cm_read;
    cuse_write_t *cm_write;
    cuse_ioctl_t *cm_ioctl;
    cuse_poll_t *cm_poll;
};
```

## HISTORY

**libcuse** was written by Hans Petter Selasky.