

CDI Nachschlagewerk

0.2

Erzeugt von Doxygen 1.5.4

Fri Jan 2 16:59:50 2009

Inhaltsverzeichnis

1	CDI Modul-Verzeichnis	1
1.1	CDI Module	1
2	CDI Datenstruktur-Verzeichnis	3
2.1	CDI Datenstrukturen	3
3	CDI Datei-Verzeichnis	5
3.1	CDI Auflistung der Dateien	5
4	CDI Modul-Dokumentation	7
4.1	Fs	7
4.2	Net	8
4.3	Scsi	9
4.4	Storage	10
4.5	Core	11
5	CDI Datenstruktur-Dokumentation	13
5.1	cdi_bios_memory Strukturreferenz	13
5.2	cdi_bios_registers Strukturreferenz	15
5.3	cdi_cache Strukturreferenz	17
5.4	cdi_cache_block Strukturreferenz	18
5.5	cdi_device Strukturreferenz	19
5.6	cdi_dma_handle Strukturreferenz	20
5.7	cdi_driver Strukturreferenz	21
5.8	cdi_fs_acl_entry Strukturreferenz	22
5.9	cdi_fs_acl_entry_grp_num Strukturreferenz	23
5.10	cdi_fs_acl_entry_grp_str Strukturreferenz	24
5.11	cdi_fs_acl_entry_usr_num Strukturreferenz	25
5.12	cdi_fs_acl_entry_usr_str Strukturreferenz	26
5.13	cdi_fs_driver Strukturreferenz	27

5.14	cdi_fs_filesystem Strukturreferenz	28
5.15	cdi_fs_res Strukturreferenz	30
5.16	cdi_fs_res_dir Strukturreferenz	33
5.17	cdi_fs_res_file Strukturreferenz	34
5.18	cdi_fs_res_flags Strukturreferenz	36
5.19	cdi_fs_res_link Strukturreferenz	38
5.20	cdi_fs_res_res Strukturreferenz	39
5.21	cdi_fs_res_special Strukturreferenz	43
5.22	cdi_fs_stream Strukturreferenz	44
5.23	cdi_net_device Strukturreferenz	45
5.24	cdi_net_driver Strukturreferenz	46
5.25	cdi_pci_device Strukturreferenz	47
5.26	cdi_pci_resource Strukturreferenz	49
5.27	cdi_scsi_device Strukturreferenz	50
5.28	cdi_scsi_driver Strukturreferenz	51
5.29	cdi_storage_device Strukturreferenz	52
5.30	cdi_storage_driver Strukturreferenz	53
6	CDI Datei-Dokumentation	55
6.1	bios.h -Dateireferenz	55
6.2	cache.h -Dateireferenz	56
6.3	cdi.h -Dateireferenz	59
6.4	dma.h -Dateireferenz	61
6.5	fs.h -Dateireferenz	63
6.6	io.h -Dateireferenz	68
6.7	lists.h -Dateireferenz	69
6.8	misc.h -Dateireferenz	72
6.9	net.h -Dateireferenz	74
6.10	pci.h -Dateireferenz	75
6.11	scsi.h -Dateireferenz	77
6.12	storage.h -Dateireferenz	78

Kapitel 1

CDI Modul-Verzeichnis

1.1 CDI Module

Hier folgt die Aufzählung aller Module:

Fs	7
Net	8
Scsi	9
Storage	10
Core	11

Kapitel 2

CDI Datenstruktur-Verzeichnis

2.1 CDI Datenstrukturen

Hier folgt die Aufzählung aller Datenstrukturen mit einer Kurzbeschreibung:

cdi_bios_memory	13
cdi_bios_registers	15
cdi_cache	17
cdi_cache_block	18
cdi_device	19
cdi_dma_handle	20
cdi_driver	21
cdi_fs_acl_entry	22
cdi_fs_acl_entry_grp_num	23
cdi_fs_acl_entry_grp_str	24
cdi_fs_acl_entry_usr_num	25
cdi_fs_acl_entry_usr_str	26
cdi_fs_driver	27
cdi_fs_filesystem	28
cdi_fs_res	30
cdi_fs_res_dir	33
cdi_fs_res_file	34
cdi_fs_res_flags	36
cdi_fs_res_link	38
cdi_fs_res_res	39
cdi_fs_res_special	43
cdi_fs_stream	44
cdi_net_device	45
cdi_net_driver	46
cdi_pci_device	47
cdi_pci_resource	49
cdi_scsi_device	50
cdi_scsi_driver	51
cdi_storage_device	52
cdi_storage_driver	53

Kapitel 3

CDI Datei-Verzeichnis

3.1 CDI Auflistung der Dateien

Hier folgt die Aufzählung aller Dateien mit einer Kurzbeschreibung:

bios.h	55
cache.h	56
cdi.h	59
dma.h	61
fs.h	63
io.h	68
lists.h	69
misc.h	72
net.h	74
pci.h	75
scsi.h	77
storage.h	78

Kapitel 4

CDI Modul-Dokumentation

4.1 Fs

Dateisystemtreiber in CDI

4.2 Net

Treiber fuer Netzwerkkarten

4.3 Scsi

Treiber fuer SCSI-Hosts

4.4 Storage

Treiber fuer Massenspeichergeraete

4.5 Core

Kapitel 5

CDI Datenstruktur-Dokumentation

5.1 cdi_bios_memory Strukturreferenz

```
#include <bios.h>
```

Datenfelder

- `uintptr_t dest`
- `void * src`
- `uint16_t size`

5.1.1 Ausführliche Beschreibung

Struktur zum Zugriff auf Speicherbereiche des 16bit-Prozesses

Definiert in Zeile 32 der Datei bios.h.

5.1.2 Dokumentation der Datenelemente

5.1.2.1 `uintptr_t cdi_bios_memory::dest`

Virtuelle Adresse im Speicher des 16bit-Prozesses. Muss unterhalb von 0xC0000 liegen.

Definiert in Zeile 37 der Datei bios.h.

5.1.2.2 `void* cdi_bios_memory::src`

Pointer auf reservierten Speicher für die Daten des Speicherbereichs. Wird beim Start des Tasks zum Initialisieren des Bereichs benutzt und erhält auch wieder die Daten nach dem BIOS-Aufruf.

Definiert in Zeile 43 der Datei bios.h.

5.1.2.3 `uint16_t cdi_bios_memory::size`

Laenge des Speicherbereichs

Definiert in Zeile 47 der Datei bios.h.

Die Dokumentation für diese Struktur wurde erzeugt aufgrund der Datei:

- [bios.h](#)

5.2 cdi_bios_registers Strukturreferenz

```
#include <bios.h>
```

Datenfelder

- [uint16_t ax](#)
- [uint16_t bx](#)
- [uint16_t cx](#)
- [uint16_t dx](#)
- [uint16_t si](#)
- [uint16_t di](#)
- [uint16_t ds](#)
- [uint16_t es](#)

5.2.1 Ausführliche Beschreibung

Definiert in Zeile 18 der Datei bios.h.

5.2.2 Dokumentation der Datenelemente

5.2.2.1 `uint16_t cdi_bios_registers::ax`

Definiert in Zeile 19 der Datei bios.h.

5.2.2.2 `uint16_t cdi_bios_registers::bx`

Definiert in Zeile 20 der Datei bios.h.

5.2.2.3 `uint16_t cdi_bios_registers::cx`

Definiert in Zeile 21 der Datei bios.h.

5.2.2.4 `uint16_t cdi_bios_registers::dx`

Definiert in Zeile 22 der Datei bios.h.

5.2.2.5 `uint16_t cdi_bios_registers::si`

Definiert in Zeile 23 der Datei bios.h.

5.2.2.6 `uint16_t cdi_bios_registers::di`

Definiert in Zeile 24 der Datei bios.h.

5.2.2.7 uint16_t cdi_bios_registers::ds

Definiert in Zeile 25 der Datei bios.h.

5.2.2.8 uint16_t cdi_bios_registers::es

Definiert in Zeile 26 der Datei bios.h.

Die Dokumentation für diese Struktur wurde erzeugt aufgrund der Datei:

- [bios.h](#)

5.3 cdi_cache Strukturreferenz

```
#include <cache.h>
```

Datenfelder

- [size_t block_size](#)

5.3.1 Ausführliche Beschreibung

Definiert in Zeile 16 der Datei cache.h.

5.3.2 Dokumentation der Datenelemente

5.3.2.1 `size_t cdi_cache::block_size`

Groesse der Blocks, die der Cache verwaltet

Definiert in Zeile 18 der Datei cache.h.

Die Dokumentation für diese Struktur wurde erzeugt aufgrund der Datei:

- [cache.h](#)

5.4 cdi_cache_block Strukturreferenz

```
#include <cache.h>
```

Datenfelder

- `uint64_t number`
- `void * data`
- `void * private`

5.4.1 Ausführliche Beschreibung

Block fuer direkten, schnellen Zugriff.

Definiert in Zeile 24 der Datei cache.h.

5.4.2 Dokumentation der Datenelemente

5.4.2.1 `uint64_t cdi_cache_block::number`

Blocknummer

Definiert in Zeile 26 der Datei cache.h.

5.4.2.2 `void* cdi_cache_block::data`

Zeiger auf die Daten

Definiert in Zeile 29 der Datei cache.h.

5.4.2.3 `void* cdi_cache_block::private`

Pointer auf `blkpriv_len` Bytes, die vom Aufrufer benutzt werden koennen

Definiert in Zeile 32 der Datei cache.h.

Die Dokumentation für diese Struktur wurde erzeugt aufgrund der Datei:

- [cache.h](#)

5.5 cdi_device Strukturreferenz

```
#include <cdi.h>
```

Datenfelder

- [cdi_device_type_t](#) type
- const char * [name](#)
- struct [cdi_driver](#) * [driver](#)

5.5.1 Ausführliche Beschreibung

Definiert in Zeile 30 der Datei cdi.h.

5.5.2 Dokumentation der Datenelemente

5.5.2.1 [cdi_device_type_t](#) [cdi_device::type](#)

Definiert in Zeile 31 der Datei cdi.h.

5.5.2.2 [const char*](#) [cdi_device::name](#)

Definiert in Zeile 32 der Datei cdi.h.

5.5.2.3 [struct cdi_driver*](#) [cdi_device::driver](#) [read]

Definiert in Zeile 33 der Datei cdi.h.

Die Dokumentation für diese Struktur wurde erzeugt aufgrund der Datei:

- [cdi.h](#)

5.6 cdi_dma_handle Strukturreferenz

```
#include <dma.h>
```

Datenfelder

- uint8_t [channel](#)
- size_t [length](#)
- uint8_t [mode](#)
- void * [buffer](#)
- FILE * [file](#)

5.6.1 Ausführliche Beschreibung

Definiert in Zeile 18 der Datei dma.h.

5.6.2 Dokumentation der Datenelemente

5.6.2.1 uint8_t cdi_dma_handle::channel

Definiert in Zeile 19 der Datei dma.h.

5.6.2.2 size_t cdi_dma_handle::length

Definiert in Zeile 20 der Datei dma.h.

5.6.2.3 uint8_t cdi_dma_handle::mode

Definiert in Zeile 21 der Datei dma.h.

5.6.2.4 void* cdi_dma_handle::buffer

Definiert in Zeile 22 der Datei dma.h.

5.6.2.5 FILE* cdi_dma_handle::file

Definiert in Zeile 25 der Datei dma.h.

Die Dokumentation für diese Struktur wurde erzeugt aufgrund der Datei:

- [dma.h](#)

5.7 cdi_driver Strukturreferenz

```
#include <cdi.h>
```

Datenfelder

- [cdi_device_type_t](#) type
- const char * [name](#)
- [cdi_list_t](#) devices
- void(* [init_device](#))(struct [cdi_device](#) *device)
- void(* [remove_device](#))(struct [cdi_device](#) *device)
- void(* [destroy](#))(struct [cdi_driver](#) *driver)

5.7.1 Ausführliche Beschreibung

Definiert in Zeile 36 der Datei cdi.h.

5.7.2 Dokumentation der Datenelemente

5.7.2.1 [cdi_device_type_t](#) [cdi_driver::type](#)

Definiert in Zeile 37 der Datei cdi.h.

5.7.2.2 [const char*](#) [cdi_driver::name](#)

Definiert in Zeile 38 der Datei cdi.h.

5.7.2.3 [cdi_list_t](#) [cdi_driver::devices](#)

Definiert in Zeile 39 der Datei cdi.h.

5.7.2.4 [void\(* cdi_driver::init_device\)\(struct cdi_device *device\)](#)

5.7.2.5 [void\(* cdi_driver::remove_device\)\(struct cdi_device *device\)](#)

5.7.2.6 [void\(* cdi_driver::destroy\)\(struct cdi_driver *driver\)](#)

Die Dokumentation für diese Struktur wurde erzeugt aufgrund der Datei:

- [cdi.h](#)

5.8 cdi_fs_acl_entry Strukturreferenz

```
#include <fs.h>
```

Datenfelder

- [cdi_fs_acl_entry_type_t](#) type
- struct [cdi_fs_res_flags](#) flags

5.8.1 Ausführliche Beschreibung

Der Basiseintrag in einer ACL, von dem die anderen Typen der Einträge abgeleitet sind.

Definiert in Zeile 579 der Datei fs.h.

5.8.2 Dokumentation der Datenelemente

5.8.2.1 cdi_fs_acl_entry_type_t cdi_fs_acl_entry::type

Typ des Eintrages, eine der obigen Konstanten

Definiert in Zeile 581 der Datei fs.h.

5.8.2.2 struct cdi_fs_res_flags cdi_fs_acl_entry::flags [read]

Flags

Definiert in Zeile 584 der Datei fs.h.

Die Dokumentation für diese Struktur wurde erzeugt aufgrund der Datei:

- [fs.h](#)

5.9 cdi_fs_acl_entry_grp_num Strukturreferenz

```
#include <fs.h>
```

Datenfelder

- struct [cdi_fs_acl_entry](#) entry
- gid_t [group_id](#)

5.9.1 Ausführliche Beschreibung

Definiert in Zeile 606 der Datei fs.h.

5.9.2 Dokumentation der Datenelemente

5.9.2.1 struct cdi_fs_acl_entry cdi_fs_acl_entry_grp_num::entry [read]

Definiert in Zeile 607 der Datei fs.h.

5.9.2.2 gid_t cdi_fs_acl_entry_grp_num::group_id

Gruppen-ID

Definiert in Zeile 610 der Datei fs.h.

Die Dokumentation für diese Struktur wurde erzeugt aufgrund der Datei:

- [fs.h](#)

5.10 `cdi_fs_acl_entry_grp_str` Strukturreferenz

```
#include <fs.h>
```

Datenfelder

- struct `cdi_fs_acl_entry` `entry`
- char * `group_name`

5.10.1 Ausführliche Beschreibung

Definiert in Zeile 613 der Datei `fs.h`.

5.10.2 Dokumentation der Datenelemente

5.10.2.1 `struct cdi_fs_acl_entry cdi_fs_acl_entry_grp_str::entry` [read]

Definiert in Zeile 614 der Datei `fs.h`.

5.10.2.2 `char* cdi_fs_acl_entry_grp_str::group_name`

Gruppenname

Definiert in Zeile 617 der Datei `fs.h`.

Die Dokumentation für diese Struktur wurde erzeugt aufgrund der Datei:

- [fs.h](#)

5.11 cdi_fs_acl_entry_usr_num Strukturreferenz

```
#include <fs.h>
```

Datenfelder

- struct [cdi_fs_acl_entry](#) entry
- uid_t [user_id](#)

5.11.1 Ausführliche Beschreibung

Eintraege fuer die einzelnen ACL-Eintragstypen

Siehe auch:

struct [cdi_fs_acl_entry](#)

Definiert in Zeile 592 der Datei fs.h.

5.11.2 Dokumentation der Datenelemente

5.11.2.1 struct cdi_fs_acl_entry cdi_fs_acl_entry_usr_num::entry [read]

Definiert in Zeile 593 der Datei fs.h.

5.11.2.2 uid_t cdi_fs_acl_entry_usr_num::user_id

Benutzer-ID

Definiert in Zeile 596 der Datei fs.h.

Die Dokumentation für diese Struktur wurde erzeugt aufgrund der Datei:

- [fs.h](#)

5.12 `cdi_fs_acl_entry_usr_str` Strukturreferenz

```
#include <fs.h>
```

Datenfelder

- struct `cdi_fs_acl_entry` `entry`
- char * `user_name`

5.12.1 Ausführliche Beschreibung

Definiert in Zeile 599 der Datei `fs.h`.

5.12.2 Dokumentation der Datenelemente

5.12.2.1 `struct cdi_fs_acl_entry cdi_fs_acl_entry_usr_str::entry` [read]

Definiert in Zeile 600 der Datei `fs.h`.

5.12.2.2 `char* cdi_fs_acl_entry_usr_str::user_name`

Benutzername

Definiert in Zeile 603 der Datei `fs.h`.

Die Dokumentation für diese Struktur wurde erzeugt aufgrund der Datei:

- [fs.h](#)

5.13 cdi_fs_driver Strukturreferenz

```
#include <fs.h>
```

Datenfelder

- struct [cdi_driver drv](#)
- int(* [fs_init](#))(struct [cdi_fs_filesystem *fs](#))
- int(* [fs_destroy](#))(struct [cdi_fs_filesystem *fs](#))

5.13.1 Ausführliche Beschreibung

Diese Struktur wird fuer jeden Dateisystemtreiber einmal erstellt

Definiert in Zeile 32 der Datei fs.h.

5.13.2 Dokumentation der Datenelemente

5.13.2.1 struct cdi_driver cdi_fs_driver::drv [read]

Definiert in Zeile 33 der Datei fs.h.

5.13.2.2 int(* cdi_fs_driver::fs_init)(struct cdi_fs_filesystem *fs)

Neues Dateisystem initialisieren; Diese Funktion muss das root_object in der Dateisystemstruktur eintragen.

Rückgabe:

Wenn das Dateisystem erfolgreich initialisiert wurde 1, sonst 0. Falls ein Fehler auftritt, muss das error-Feld in der Dateisystemstruktur gesetzt werden.

5.13.2.3 int(* cdi_fs_driver::fs_destroy)(struct cdi_fs_filesystem *fs)

Dateisystem deinitialisieren

Rückgabe:

Wenn das Dateisystem erfolgreich deinitialisiert wurde 1, 0 sonst. Falls ein Fehler auftritt, muss das error-Feld in der Dateisystemstruktur gesetzt werden.

Die Dokumentation für diese Struktur wurde erzeugt aufgrund der Datei:

- [fs.h](#)

5.14 cdi_fs_filesystem Strukturreferenz

```
#include <fs.h>
```

Datenfelder

- struct [cdi_fs_driver](#) * [driver](#)
- struct [cdi_fs_res](#) * [root_res](#)
- int [error](#)
- int [read_only](#)
- FILE * [device](#)
- void * [opaque](#)

5.14.1 Ausführliche Beschreibung

Diese Struktur wird fuer jedes eingebundene Dateisystem einmal erstellt.

Definiert in Zeile 59 der Datei fs.h.

5.14.2 Dokumentation der Datenelemente

5.14.2.1 struct [cdi_fs_driver](#)* [cdi_fs_filesystem::driver](#) [read]

Treiber, dem das Dateisystem gehoert

Definiert in Zeile 62 der Datei fs.h.

5.14.2.2 struct [cdi_fs_res](#)* [cdi_fs_filesystem::root_res](#) [read]

Wurzelverzeichnis des Dateisystems

Definiert in Zeile 65 der Datei fs.h.

5.14.2.3 int [cdi_fs_filesystem::error](#)

Falls ein gravierender Fehler auftritt, wird diese Fehlernummer gesetzt. Wenn sie != 0 ist wird das Dateisystem fuer Schreibzugriffe gesperrt.

Definiert in Zeile 71 der Datei fs.h.

5.14.2.4 int [cdi_fs_filesystem::read_only](#)

Das Dateisystem darf nicht geschrieben werden. Damit schlaegt unter anderem [cdi_fs_write_data](#) fehl.

Definiert in Zeile 77 der Datei fs.h.

5.14.2.5 FILE* [cdi_fs_filesystem::device](#)

OS-spezifisch: Deskriptor fuer den Datentraeger

Definiert in Zeile 86 der Datei fs.h.

5.14.2.6 void* `cdi_fs_filesystem::opaque`

Zeiger den der Treiber fuer eigene Daten zum Dateisystem benutzen kann

Definiert in Zeile 91 der Datei `fs.h`.

Die Dokumentation für diese Struktur wurde erzeugt aufgrund der Datei:

- [fs.h](#)

5.15 cdi_fs_res Strukturreferenz

```
#include <fs.h>
```

Datenfelder

- char * [name](#)
 - [cdi_fs_lock_t](#) lock
 - int loaded
 - int stream_cnt
 - struct [cdi_fs_res](#) * parent
 - [cdi_list_t](#) children
 - char * link_path
 - [cdi_list_t](#) acl
 - struct [cdi_fs_res_flags](#) flags
 - [cdi_fs_res_type_t](#) type
-
- struct [cdi_fs_res_res](#) * res
 - struct [cdi_fs_res_file](#) * file
 - struct [cdi_fs_res_dir](#) * dir
 - struct [cdi_fs_res_link](#) * link
 - struct [cdi_fs_res_special](#) * special

5.15.1 Ausführliche Beschreibung

Das Dateisystem wird hier nur mit abstrakten Strukturen vom Typ [cdi_fs_res](#) dargestellt. Diese können beispielsweise sowohl reguläre Datei als auch Verzeichnis gleichzeitig darstellen.

Weiter gilt, dass Ressourcen, die zu keiner Klasse gehören, nicht persistent sind.

Definiert in Zeile 228 der Datei fs.h.

5.15.2 Dokumentation der Datenelemente

5.15.2.1 char* cdi_fs_res::name

Name der Ressource

Definiert in Zeile 230 der Datei fs.h.

5.15.2.2 cdi_fs_lock_t cdi_fs_res::lock

Lock fuer diese Ressource

Definiert in Zeile 233 der Datei fs.h.

5.15.2.3 int cdi_fs_res::loaded

Flag ob die Ressource geladen ist(1) oder nicht(0). Ist sie danicht, muss nur name und res definiert sein. In res darf nur load aufgerufen werden.

Definiert in Zeile 240 der Datei fs.h.

5.15.2.4 int cdi_fs_res::stream_cnt

Referenzzähler fuer Implementation. Muss beim erstellen der Ressource mit 0 initialisiert werden
Definiert in Zeile 246 der Datei fs.h.

5.15.2.5 struct cdi_fs_res* cdi_fs_res::parent [read]

Verweis auf das Elternobjekt
Definiert in Zeile 250 der Datei fs.h.

5.15.2.6 cdi_list_t cdi_fs_res::children

Liste mit allfaelligen Kindobjekten
Definiert in Zeile 253 der Datei fs.h.

5.15.2.7 char* cdi_fs_res::link_path

Link-Pfad
Definiert in Zeile 257 der Datei fs.h.

5.15.2.8 cdi_list_t cdi_fs_res::acl

ACL

Siehe auch:

struct [cdi_fs_acl_entry](#)

Definiert in Zeile 264 der Datei fs.h.

5.15.2.9 struct cdi_fs_res_flags cdi_fs_res::flags [read]

Flags
Definiert in Zeile 267 der Datei fs.h.

5.15.2.10 struct cdi_fs_res_res* cdi_fs_res::res [read]

Einzelne Klassen, zu denen die Ressourcen gehoeren kann, oder Null falls es zu einer Bestimmten Klasse nicht gehoert.

Definiert in Zeile 275 der Datei fs.h.

5.15.2.11 struct cdi_fs_res_file* cdi_fs_res::file [read]

Einzelne Klassen, zu denen die Ressourcen gehoeren kann, oder Null falls es zu einer Bestimmten Klasse nicht gehoert.

Definiert in Zeile 276 der Datei fs.h.

5.15.2.12 struct cdi_fs_res_dir* cdi_fs_res::dir [read]

Einzelne Klassen, zu denen die Ressourcen gehoeren kann, oder Null falls es zu einer Bestimmten Klasse nicht gehoert.

Definiert in Zeile 277 der Datei fs.h.

5.15.2.13 struct cdi_fs_res_link* cdi_fs_res::link [read]

Einzelne Klassen, zu denen die Ressourcen gehoeren kann, oder Null falls es zu einer Bestimmten Klasse nicht gehoert.

Definiert in Zeile 278 der Datei fs.h.

5.15.2.14 struct cdi_fs_res_special* cdi_fs_res::special [read]

Einzelne Klassen, zu denen die Ressourcen gehoeren kann, oder Null falls es zu einer Bestimmten Klasse nicht gehoert.

Definiert in Zeile 279 der Datei fs.h.

5.15.2.15 cdi_fs_res_type_t cdi_fs_res::type

Falls die Ressource zu einer Spezialklasse gehoert, wird hier angegeben, um welchen Typ von Spezialresource sie gehoert.

Definiert in Zeile 286 der Datei fs.h.

Die Dokumentation für diese Struktur wurde erzeugt aufgrund der Datei:

- [fs.h](#)

5.16 `cdi_fs_res_dir` Strukturreferenz

```
#include <fs.h>
```

Datenfelder

- `cdi_list_t(* list)(struct cdi_fs_stream *stream)`
- `int(* create_child)(struct cdi_fs_stream *stream, const char *name, struct cdi_fs_res *parent)`

5.16.1 Ausführliche Beschreibung

Definiert in Zeile 475 der Datei `fs.h`.

5.16.2 Dokumentation der Datenelemente

5.16.2.1 `cdi_list_t(* cdi_fs_res_dir::list)(struct cdi_fs_stream *stream)`

Diese Funktion gibt einen Pointer auf die Liste mit den Einträgen zurück. Hier wird nicht einfach fix der Pointer in `fs_res` genommen, damit dort auch "versteckte" Einträge vorhanden sein könnten. (Ich meine hier nicht irgend ein versteckt-Flag dass die Dateien vor dem Benutzer Verstecken soll, sondern eher von fuer den Internen gebrauch angelegten Einträgen. Diese Liste muss nicht vom Aufrufer freigegeben werden, da einige Treiber hier direkt children aus `fs_res` benutzen, und andere dafuer eine eigene Liste erstellen, die sie intern abspeichern.

Parameter:

stream Stream

Rückgabe:

Pointer auf eine Liste mit den Untereinträgen vom Typ `cdi_fs_res`.

5.16.2.2 `int(* cdi_fs_res_dir::create_child)(struct cdi_fs_stream *stream, const char *name, struct cdi_fs_res *parent)`

Neue Ressource in der Aktuellen erstellen. Diese wird erstmal noch keiner Klasse zugewiesen. Diese Funktion wird mit einem NULL-Pointer als Ressource im Stream aufgerufen. Dieser NULL-Pointer muss bei erfolgreichem Beenden durch einen Pointer auf die neue Ressource ersetzt worden sein.

Parameter:

stream Mit NULL-Pointer als Ressource

name Name der neuen Ressource

parent Ressource, der die neue Ressource als Kindressource zugeordnet werden soll.

Rückgabe:

Falls die Ressource erfolgreich erstellt wurde 1, sonst 0

Die Dokumentation für diese Struktur wurde erzeugt aufgrund der Datei:

- [fs.h](#)

5.17 cdi_fs_res_file Strukturreferenz

```
#include <fs.h>
```

Datenfelder

- int `executable`
- `size_t(* read)(struct cdi_fs_stream *stream, uint64_t start, size_t size, void *buffer)`
- `size_t(* write)(struct cdi_fs_stream *stream, uint64_t start, size_t size, const void *buffer)`
- `int(* truncate)(struct cdi_fs_stream *stream, uint64_t size)`

5.17.1 Ausführliche Beschreibung

Definiert in Zeile 420 der Datei fs.h.

5.17.2 Dokumentation der Datenelemente

5.17.2.1 int cdi_fs_res_file::executable

Dateien in dieser Klasse sind grundsätzlich ausführbar, wenn in der Flag-Struktur in der Ressource nicht anders angegeben

Definiert in Zeile 423 der Datei fs.h.

5.17.2.2 size_t(* cdi_fs_res_file::read)(struct cdi_fs_stream *stream, uint64_t start, size_t size, void *buffer)

Daten aus dieser Datei lesen. Wird nur aufgerufen, wenn es durch die Flags oder Berechtigungen nicht verhindert wird.

Im Fehlerfall wird je nach Fehler die Fehlernummer im Handle und die im Device gesetzt.

Parameter:

- stream* Stream
- start* Position von der an gelesen werden soll
- size* Groesse der zu lesenden Daten
- buffer* Puffer in den die Daten gelsen werden sollen

Rückgabe:

Gelesene Bytes, oder 0 im Fehlerfall

5.17.2.3 size_t(* cdi_fs_res_file::write)(struct cdi_fs_stream *stream, uint64_t start, size_t size, const void *buffer)

Daten in diese Datei schreiben. Wird nur aufgerufen, wenn es durch die Flags oder Berechtigungen nicht verhindert wird.

Im Fehlerfall wird je nach Fehler die Fehlernummer im Handle und die im Device gesetzt.

Parameter:

- stream* Stream
- start* Position an die geschrieben werden soll
- size* Groesse der zu schreibenden Daten
- buffer* Puffer aus dem die Daten gelesen werden sollen

Rückgabe:

Geschriebene Bytes oder 0 im Fehlerfall

5.17.2.4 `int(* cdi_fs_res_file::truncate)(struct cdi_fs_stream *stream, uint64_t size)`

Groesse der Datei anpassen. Diese Funktion kann sowohl fuers Vergroessern, als auch fuers Verkleinern benutzt werden.

Im Fehlerfall wird je nach Fehler die Fehlernummer im Handle und die im Device gesetzt.

Parameter:

- stream* Stream
- size* Neue Groesse der Datei

Rückgabe:

1 bei Erfolg, im Fehlerfall 0

Die Dokumentation für diese Struktur wurde erzeugt aufgrund der Datei:

- [fs.h](#)

5.18 cdi_fs_res_flags Strukturreferenz

```
#include <fs.h>
```

Datenfelder

- int [remove](#)
- int [rename](#)
- int [move](#)
- int [read](#)
- int [write](#)
- int [execute](#)
- int [browse](#)
- int [read_link](#)
- int [write_link](#)
- int [create_child](#)

5.18.1 Ausführliche Beschreibung

Siese Struktur stellt die Moeglichkeiten, die an einer Ressource zur Verfuegung stehen, dar.
Definiert in Zeile 159 der Datei fs.h.

5.18.2 Dokumentation der Datenelemente

5.18.2.1 int cdi_fs_res_flags::remove

Ressource loeschen

Definiert in Zeile 161 der Datei fs.h.

5.18.2.2 int cdi_fs_res_flags::rename

Ressource umbenennen

Definiert in Zeile 163 der Datei fs.h.

5.18.2.3 int cdi_fs_res_flags::move

Ressource verschieben

Definiert in Zeile 165 der Datei fs.h.

5.18.2.4 int cdi_fs_res_flags::read

Lesender Zugriff gestattet

Definiert in Zeile 167 der Datei fs.h.

5.18.2.5 int cdi_fs_res_flags::write

Schreibender Zugriff gestattet

Definiert in Zeile 169 der Datei fs.h.

5.18.2.6 int cdi_fs_res_flags::execute

Ausfuehren gestattet

Definiert in Zeile 171 der Datei fs.h.

5.18.2.7 int cdi_fs_res_flags::browse

Auflisten der Verzeichniseintraege gestattet

Definiert in Zeile 173 der Datei fs.h.

5.18.2.8 int cdi_fs_res_flags::read_link

Aufloesen des Links

Definiert in Zeile 175 der Datei fs.h.

5.18.2.9 int cdi_fs_res_flags::write_link

Aendern des Links

Definiert in Zeile 177 der Datei fs.h.

5.18.2.10 int cdi_fs_res_flags::create_child

Anlegen eines Untereintrags

Definiert in Zeile 179 der Datei fs.h.

Die Dokumentation für diese Struktur wurde erzeugt aufgrund der Datei:

- [fs.h](#)

5.19 cdi_fs_res_link Strukturreferenz

```
#include <fs.h>
```

Datenfelder

- `const char *(* read_link)(struct cdi_fs_stream *stream)`
- `int(* write_link)(struct cdi_fs_stream *stream, const char *path)`

5.19.1 Ausführliche Beschreibung

Definiert in Zeile 512 der Datei fs.h.

5.19.2 Dokumentation der Datenelemente

5.19.2.1 `const char *(* cdi_fs_res_link::read_link)(struct cdi_fs_stream *stream)`

Diese Funktion liest den Pfad aus, auf den der Link zeigt

Parameter:

stream Stream

Rückgabe:

Pointer auf einen Buffer der den Pfad beinhaltet. Dieses Puffer darf vom Aufrufer nicht veraendert werden.

5.19.2.2 `int(* cdi_fs_res_link::write_link)(struct cdi_fs_stream *stream, const char *path)`

Aendert den Pfad auf den der Link zeigt

Parameter:

stream Stream

path Neuer Pfad

Rückgabe:

Wenn der Link erfolgreich geschrieben wurde 1, 0 sonst

Die Dokumentation für diese Struktur wurde erzeugt aufgrund der Datei:

- [fs.h](#)

5.20 cdi_fs_res_res Strukturreferenz

```
#include <fs.h>
```

Datenfelder

- `int(* load)(struct cdi_fs_stream *stream)`
- `int(* unload)(struct cdi_fs_stream *stream)`
- `int(* remove)(struct cdi_fs_stream *stream)`
- `int(* rename)(struct cdi_fs_stream *stream, const char *name)`
- `int(* move)(struct cdi_fs_stream *stream, struct cdi_fs_res *dest)`
- `int(* assign_class)(struct cdi_fs_stream *stream, cdi_fs_res_class_t class)`
- `int(* remove_class)(struct cdi_fs_stream *stream, cdi_fs_res_class_t class)`
- `int64_t(* meta_read)(struct cdi_fs_stream *stream, cdi_fs_meta_t meta)`
- `int(* meta_write)(struct cdi_fs_stream *stream, cdi_fs_meta_t meta, int64_t value)`

5.20.1 Ausführliche Beschreibung

Diese Dateisystemobjekte werden in Klassen eingeteilt, die das eigentliche "Verhalten" der Ressourcen steuern. Diese Klassen beinhalten die moeglichen Operationen und auch die Eigenschaften, die fuer die Ressourcen gelten, denen diese Klassen zugeordnet sind. Das Definieren der einzelnen Klassen uebernehmen dann die einzelnen Treiber.

Die Flags koennen von der Ressource ueberschrieben werden. Es koennen allerdings nur Flags deaktiviert werden, die in der Klasse gesetzt sind un nicht umgekehrt. Das Selbe gilt auch fuer Klassen bei denen NULL-Pointer fuer Operationen eingetragen sind. Wenn zum Beispiel fuer write NULL eingetragen wird, dann bringt ein gesetztes write-Flag nichts. Diese Klasse gilt unabhaengig von den andern, also egal welche anderen Klassen angegeben sind, diese muss angegeben werden.

Definiert in Zeile 310 der Datei fs.h.

5.20.2 Dokumentation der Datenelemente

5.20.2.1 `int(* cdi_fs_res_res::load)(struct cdi_fs_stream *stream)`

Ressource laden

Parameter:

stream Stream

Rückgabe:

Falls die Ressource erfolgreich geladen wurde 1, sonst 0

5.20.2.2 `int(* cdi_fs_res_res::unload)(struct cdi_fs_stream *stream)`

Ressource entladen; Darf von der Implementation nur aufgerufen werden, wenn keine geladenen Kind-Ressourcen existieren. Das gilt aber nur fuer Verzeichnisse. Wenn andere Kind-Eintraege existieren, werden die nicht beruecksichtigt.

Parameter:

stream Stream

Rückgabe:

Falls die Ressource erfolgreich entladen wurde 1, sonst 0

5.20.2.3 int(* cdi_fs_res_res::remove)(struct cdi_fs_stream *stream)

Ressource entfernen. Diese Funktion wird nur aufgerufen, wenn die Ressource keiner Klasse mehr zugewiesen ist.

Parameter:

stream Stream

Rückgabe:

Falls die Ressource erfolgreich gelöscht wurde 1, sonst 0

5.20.2.4 int(* cdi_fs_res_res::rename)(struct cdi_fs_stream *stream, const char *name)

Namen einer Ressource ändern. Der Parameter name ist nur der Resourcennamen ohne Pfad. Zum verschieben wird [move\(\)](#) benutzt.

Parameter:

stream Stream

name Neuer Name

Rückgabe:

Falls die Ressource erfolgreich umbenannt wurde 1, sonst 0

5.20.2.5 int(* cdi_fs_res_res::move)(struct cdi_fs_stream *stream, struct cdi_fs_res *dest)

Ressource innerhalb des Dateisystems verschieben. Das Verschieben ueber Dateisystemgrenzen hinweg wird per kopieren und loeschen durchgefuehrt.

Parameter:

stream Stream

dest Pointer auf die Ressource, in die die Ressource verschoben werden soll

Rückgabe:

Falls die Ressource erfolgreich verschoben wurde 1, sonst 0

5.20.2.6 `int(* cdi_fs_res_res::assign_class)(struct cdi_fs_stream *stream, cdi_fs_res_class_t class)`

Diese Ressource einer neuen Klasse zuweisen. Diese Funktion wird nur aufgerufen, wenn die Ressource nicht bereits dieser Klasse zugewiesen ist.

Parameter:

stream Stream

class Konstante fuer den Typ der Klasse, der die Ressource zugewiesen werden soll.

Rückgabe:

1 falls die Ressource erfolgreich der Klasse zugewiesen wurde, 0 sonst

5.20.2.7 `int(* cdi_fs_res_res::remove_class)(struct cdi_fs_stream *stream, cdi_fs_res_class_t class)`

Diese Ressource aus einer Klasse entfernen. Diese Funktion wird nur aufgerufen, wenn die Ressource zu dieser Klasse gehoert.

Bei Verzeichnissen muss von der Implementierung garantiert werden, dass diese Funktion nicht aufgerufen wird, solange das Verzeichnis noch Kindressourcen hat.

Parameter:

class Konstante fuer den Typ der Klasse, aus der die Ressource entfernt werden soll.

Rückgabe:

1 falls die Ressource erfolgreich aus der Klasse entfernt wurde, 0 sonst

5.20.2.8 `int64_t(* cdi_fs_res_res::meta_read)(struct cdi_fs_stream *stream, cdi_fs_meta_t meta)`

Metaeigenschaft lesen

Parameter:

stream Stream

meta Konstante fuer die gewuenschte Metaeigenschaft

Rückgabe:

Wert der Metaeigenschaft

5.20.2.9 `int(* cdi_fs_res_res::meta_write)(struct cdi_fs_stream *stream, cdi_fs_meta_t meta, int64_t value)`

Metaeigenschaft schreiben

Parameter:

stream Stream

meta Konstante fuer die gewuenschte Metaeigenschaft

value Neuen Wert fuer die Metaeigenschaft

Rückgabe:

Falls die Metaeigenschaft erfolgreich geaendert wurde 1, sonst 0

Die Dokumentation für diese Struktur wurde erzeugt aufgrund der Datei:

- [fs.h](#)

5.21 cdi_fs_res_special Strukturreferenz

```
#include <fs.h>
```

Datenfelder

- `int(* dev_read)(struct cdi_fs_stream *stream, dev_t *dev)`
- `int(* dev_write)(struct cdi_fs_stream *stream, dev_t dev)`

5.21.1 Ausführliche Beschreibung

Definiert in Zeile 534 der Datei fs.h.

5.21.2 Dokumentation der Datenelemente

5.21.2.1 `int(* cdi_fs_res_special::dev_read)(struct cdi_fs_stream *stream, dev_t *dev)`

Geraeteadresse der Spezialdatei Lesen

Parameter:

stream Stream

dev Pointer auf die Variable in der die Geraeteadresse gespeichert werden soll.

Rückgabe:

Falls die Geraeteadresse erfolgreich gelesen wurde 1, sonst 0

5.21.2.2 `int(* cdi_fs_res_special::dev_write)(struct cdi_fs_stream *stream, dev_t dev)`

Geraeteadresse der Spezialdatei Aendern

Parameter:

stream Stream

dev Die neue Geraeteadresse

Rückgabe:

Falls die Geraeteadresse erfolgreich geaendert wurde 1, sonst 0

Die Dokumentation für diese Struktur wurde erzeugt aufgrund der Datei:

- [fs.h](#)

5.22 cdi_fs_stream Strukturreferenz

```
#include <fs.h>
```

Datenfelder

- struct [cdi_fs_filesystem](#) * fs
- struct [cdi_fs_res](#) * res
- [cdi_fs_error_t](#) error

5.22.1 Ausführliche Beschreibung

Der Stream stellt die Verbindung zwischen Aufrufer und Ressource dar.

Definiert in Zeile 121 der Datei fs.h.

5.22.2 Dokumentation der Datenelemente

5.22.2.1 struct [cdi_fs_filesystem](#)* [cdi_fs_stream::fs](#) [read]

Dateisystem

Definiert in Zeile 123 der Datei fs.h.

5.22.2.2 struct [cdi_fs_res](#)* [cdi_fs_stream::res](#) [read]

Betroffene Ressource

Definiert in Zeile 126 der Datei fs.h.

5.22.2.3 [cdi_fs_error_t](#) [cdi_fs_stream::error](#)

Fehlernummer

Definiert in Zeile 129 der Datei fs.h.

Die Dokumentation für diese Struktur wurde erzeugt aufgrund der Datei:

- [fs.h](#)

5.23 cdi_net_device Strukturreferenz

```
#include <net.h>
```

Datenfelder

- struct [cdi_device dev](#)
- uint64_t [mac](#): 48
- int [number](#)
- void(* [send_packet](#))(struct [cdi_net_device](#) *device, void *data, size_t size)
- uint32_t [ip](#)

5.23.1 Ausführliche Beschreibung

Definiert in Zeile 25 der Datei net.h.

5.23.2 Dokumentation der Datenelemente

5.23.2.1 struct [cdi_device cdi_net_device::dev](#) [read]

Definiert in Zeile 26 der Datei net.h.

5.23.2.2 uint64_t [cdi_net_device::mac](#)

Definiert in Zeile 27 der Datei net.h.

5.23.2.3 int [cdi_net_device::number](#)

Definiert in Zeile 28 der Datei net.h.

5.23.2.4 void(* [cdi_net_device::send_packet](#))(struct [cdi_net_device](#) *device, void *data, size_t size)

5.23.2.5 uint32_t [cdi_net_device::ip](#)

Definiert in Zeile 34 der Datei net.h.

Die Dokumentation für diese Struktur wurde erzeugt aufgrund der Datei:

- [net.h](#)

5.24 cdi_net_driver Strukturreferenz

```
#include <net.h>
```

Datenfelder

- struct [cdi_driver drv](#)

5.24.1 Ausführliche Beschreibung

Definiert in Zeile 37 der Datei net.h.

5.24.2 Dokumentation der Datenelemente

5.24.2.1 struct cdi_driver cdi_net_driver::drv [read]

Definiert in Zeile 38 der Datei net.h.

Die Dokumentation für diese Struktur wurde erzeugt aufgrund der Datei:

- [net.h](#)

5.25 cdi_pci_device Strukturreferenz

```
#include <pci.h>
```

Datenfelder

- [uint16_t bus](#)
- [uint16_t dev](#)
- [uint16_t function](#)
- [uint16_t vendor_id](#)
- [uint16_t device_id](#)
- [uint16_t class_id](#)
- [uint8_t rev_id](#)
- [uint8_t irq](#)
- [cdi_list_t resources](#)

5.25.1 Ausführliche Beschreibung

Definiert in Zeile 19 der Datei pci.h.

5.25.2 Dokumentation der Datenelemente

5.25.2.1 uint16_t cdi_pci_device::bus

Definiert in Zeile 20 der Datei pci.h.

5.25.2.2 uint16_t cdi_pci_device::dev

Definiert in Zeile 21 der Datei pci.h.

5.25.2.3 uint16_t cdi_pci_device::function

Definiert in Zeile 22 der Datei pci.h.

5.25.2.4 uint16_t cdi_pci_device::vendor_id

Definiert in Zeile 24 der Datei pci.h.

5.25.2.5 uint16_t cdi_pci_device::device_id

Definiert in Zeile 25 der Datei pci.h.

5.25.2.6 uint16_t cdi_pci_device::class_id

Definiert in Zeile 26 der Datei pci.h.

5.25.2.7 uint8_t cdi_pci_device::rev_id

Definiert in Zeile 27 der Datei pci.h.

5.25.2.8 uint8_t cdi_pci_device::irq

Definiert in Zeile 29 der Datei pci.h.

5.25.2.9 cdi_list_t cdi_pci_device::resources

Definiert in Zeile 31 der Datei pci.h.

Die Dokumentation für diese Struktur wurde erzeugt aufgrund der Datei:

- [pci.h](#)

5.26 `cdi_pci_resource` Strukturreferenz

```
#include <pci.h>
```

Datenfelder

- `cdi_res_t` [type](#)
- `uintptr_t` [start](#)
- `size_t` [length](#)
- unsigned int [index](#)
- void * [address](#)

5.26.1 Ausführliche Beschreibung

Definiert in Zeile 39 der Datei `pci.h`.

5.26.2 Dokumentation der Datenelemente

5.26.2.1 `cdi_res_t cdi_pci_resource::type`

Definiert in Zeile 40 der Datei `pci.h`.

5.26.2.2 `uintptr_t cdi_pci_resource::start`

Definiert in Zeile 41 der Datei `pci.h`.

5.26.2.3 `size_t cdi_pci_resource::length`

Definiert in Zeile 42 der Datei `pci.h`.

5.26.2.4 `unsigned int cdi_pci_resource::index`

Definiert in Zeile 43 der Datei `pci.h`.

5.26.2.5 `void* cdi_pci_resource::address`

Definiert in Zeile 44 der Datei `pci.h`.

Die Dokumentation für diese Struktur wurde erzeugt aufgrund der Datei:

- [pci.h](#)

5.27 `cdi_scsi_device` Strukturreferenz

```
#include <scsi.h>
```

Datenfelder

- struct [cdi_device dev](#)

5.27.1 Ausführliche Beschreibung

Definiert in Zeile 24 der Datei `scsi.h`.

5.27.2 Dokumentation der Datenelemente

5.27.2.1 `struct cdi_device cdi_scsi_device::dev` [read]

Definiert in Zeile 25 der Datei `scsi.h`.

Die Dokumentation für diese Struktur wurde erzeugt aufgrund der Datei:

- [scsi.h](#)

5.28 cdi_scsi_driver Strukturreferenz

```
#include <scsi.h>
```

Datenfelder

- struct [cdi_driver drv](#)

5.28.1 Ausführliche Beschreibung

Definiert in Zeile 28 der Datei scsi.h.

5.28.2 Dokumentation der Datenelemente

5.28.2.1 struct cdi_driver cdi_scsi_driver::drv [read]

Definiert in Zeile 29 der Datei scsi.h.

Die Dokumentation für diese Struktur wurde erzeugt aufgrund der Datei:

- [scsi.h](#)

5.29 `cdi_storage_device` Strukturreferenz

```
#include <storage.h>
```

Datenfelder

- struct [cdi_device dev](#)
- size_t [block_size](#)
- uint64_t [block_count](#)

5.29.1 Ausführliche Beschreibung

Definiert in Zeile 24 der Datei `storage.h`.

5.29.2 Dokumentation der Datenelemente

5.29.2.1 `struct cdi_device cdi_storage_device::dev` [read]

Definiert in Zeile 25 der Datei `storage.h`.

5.29.2.2 `size_t cdi_storage_device::block_size`

Definiert in Zeile 26 der Datei `storage.h`.

5.29.2.3 `uint64_t cdi_storage_device::block_count`

Definiert in Zeile 27 der Datei `storage.h`.

Die Dokumentation für diese Struktur wurde erzeugt aufgrund der Datei:

- [storage.h](#)

5.30 cdi_storage_driver Strukturreferenz

```
#include <storage.h>
```

Datenfelder

- struct `cdi_driver drv`
- `int(* read_blocks)(struct cdi_storage_device *device, uint64_t start, uint64_t count, void *buffer)`
- `int(* write_blocks)(struct cdi_storage_device *device, uint64_t start, uint64_t count, void *buffer)`

5.30.1 Ausführliche Beschreibung

Definiert in Zeile 30 der Datei storage.h.

5.30.2 Dokumentation der Datenelemente

5.30.2.1 struct cdi_driver cdi_storage_driver::drv [read]

Definiert in Zeile 31 der Datei storage.h.

5.30.2.2 `int(* cdi_storage_driver::read_blocks)(struct cdi_storage_device *device, uint64_t start, uint64_t count, void *buffer)`

Liest Blocks ein

Parameter:

start Blocknummer des ersten zu lesenden Blockes (angefangen bei 0).

count Anzahl der zu lesenden Blocks

buffer Puffer in dem die Daten abgelegt werden sollen

Rückgabe:

0 bei Erfolg, -1 im Fehlerfall.

5.30.2.3 `int(* cdi_storage_driver::write_blocks)(struct cdi_storage_device *device, uint64_t start, uint64_t count, void *buffer)`

Schreibt Blocks

Parameter:

start Blocknummer des ersten zu schreibenden Blockes (angefangen bei 0).

count Anzahl der zu schreibenden Blocks

buffer Puffer aus dem die Daten gelesen werden sollen

Rückgabe:

0 bei Erfolg, -1 im Fehlerfall

Die Dokumentation für diese Struktur wurde erzeugt aufgrund der Datei:

- [storage.h](#)

Kapitel 6

CDI Datei-Dokumentation

6.1 bios.h-Dateireferenz

```
#include <stdint.h>
#include "cdi/lists.h"
```

Datenstrukturen

- struct [cdi_bios_registers](#)
- struct [cdi_bios_memory](#)

Funktionen

- int [cdi_bios_int10](#) (struct [cdi_bios_registers](#) *registers, [cdi_list_t](#) memory)

6.1.1 Dokumentation der Funktionen

6.1.1.1 int [cdi_bios_int10](#) (struct [cdi_bios_registers](#) * *registers*, [cdi_list_t](#) *memory*)

Ruft den BIOS-Interrupt 0x10 auf.

Parameter:

registers Pointer auf eine Register-Struktur. Diese wird beim Aufruf in die Register des Tasks geladen und bei Beendigung des Tasks wieder mit den Werten des Tasks gefuellt.

memory Speicherbereiche, die in den Bios-Task kopiert und bei Beendigung des Tasks wieder zurueck kopiert werden sollen. Die Liste ist vom Typ struct [cdi_bios_memory](#).

Rückgabe:

0, wenn der Aufruf erfolgreich war, -1 bei Fehlern

6.2 cache.h-Dateireferenz

```
#include <stdint.h>
```

Datenstrukturen

- struct [cdi_cache](#)
- struct [cdi_cache_block](#)

Typdefinitionen

- typedef int([cdi_cache_read_block_t](#))(struct [cdi_cache](#) *cache, uint64_t block, size_t count, void *dest, void *prv)
- typedef int([cdi_cache_write_block_t](#))(struct [cdi_cache](#) *cache, uint64_t block, size_t count, const void *src, void *prv)

Funktionen

- struct [cdi_cache](#) * [cdi_cache_create](#) (size_t block_size, size_t blkpriv_len, [cdi_cache_read_block_t](#) *read_block, [cdi_cache_write_block_t](#) *write_block, void *prv_data)
- void [cdi_cache_destroy](#) (struct [cdi_cache](#) *cache)
- int [cdi_cache_sync](#) (struct [cdi_cache](#) *cache)
- struct [cdi_cache_block](#) * [cdi_cache_block_get](#) (struct [cdi_cache](#) *cache, uint64_t blocknum, int no-read)
- void [cdi_cache_block_release](#) (struct [cdi_cache](#) *cache, struct [cdi_cache_block](#) *block)
- void [cdi_cache_block_dirty](#) (struct [cdi_cache](#) *cache, struct [cdi_cache_block](#) *block)

6.2.1 Dokumentation der benutzerdefinierten Typen

6.2.1.1 typedef int([cdi_cache_read_block_t](#))(struct [cdi_cache](#) *cache, uint64_t block, size_t count, void *dest, void *prv)

Typ fuer Cache-Callback zum einlesen eines Blocks.

Definiert in Zeile 36 der Datei cache.h.

6.2.1.2 typedef int([cdi_cache_write_block_t](#))(struct [cdi_cache](#) *cache, uint64_t block, size_t count, const void *src, void *prv)

Typ fuer Cache-Callback zum schreiben eines Blocks.

Definiert in Zeile 40 der Datei cache.h.

6.2.2 Dokumentation der Funktionen

6.2.2.1 void [cdi_cache_block_dirty](#) (struct [cdi_cache](#) * *cache*, struct [cdi_cache_block](#) * *block*)

Cache-Block als veraendert markieren

Parameter:

cache Cache-Handle

block Block-Handle

6.2.2.2 struct cdi_cache_block* cdi_cache_block_get (struct cdi_cache * cache, uint64_t blocknum, int noread) [read]

Cache-Block holen. Dabei wird intern ein Referenzzaehler erhoeht, sodass der Block nicht freigegeben wird, solange er benutzt wird. Das heisst aber auch, dass der Block nach der Benutzung wieder freigegeben werden muss, da sonst die freien Blocks ausgehen.

Parameter:

cache Cache-Handle

blocknum Blocknummer

noread Wenn != 0 wird der Block nicht eingelesen falls er noch nicht im Speicher ist. Kann benutzt werden, wenn der Block vollstaendig ueberschrieben werden soll.

Rückgabe:

Pointer auf das Block-Handle oder NULL im Fehlerfall

6.2.2.3 void cdi_cache_block_release (struct cdi_cache * cache, struct cdi_cache_block * block)

Cache-Block freigeben

Parameter:

cache Cache-Handle

block Block-Handle

6.2.2.4 struct cdi_cache* cdi_cache_create (size_t block_size, size_t blkpriv_len, cdi_cache_read_block_t * read_block, cdi_cache_write_block_t * write_block, void * prv_data) [read]

Cache erstellen

Parameter:

block_size Groesse der Blocks die der Cache verwalten soll

blkpriv_len Groesse der privaten Daten die fuer jeden Block alloziert werden und danach vom aufrufer frei benutzt werden duerfen

read_block Funktionspointer auf eine Funktion zum einlesen eines Blocks.

write_block Funktionspointer auf eine Funktion zum schreiben eines Blocks.

prv_data Wird den Callbacks als letzter Parameter uebergeben

Rückgabe:

Pointer auf das Cache-Handle

6.2.2.5 void cdi_cache_destroy (struct cdi_cache * *cache*)

Cache zerstören

6.2.2.6 int cdi_cache_sync (struct cdi_cache * *cache*)

Veraenderte Cache-Blocks auf die Platte schreiben

Rückgabe:

1 bei Erfolg, 0 im Fehlerfall

6.3 cdi.h-Dateireferenz

```
#include <stdint.h>
#include "cdi/lists.h"
```

Datenstrukturen

- struct [cdi_device](#)
- struct [cdi_driver](#)

Makrodefinitionen

- #define [CDI_STANDALONE](#)

Aufzählungen

- enum [cdi_device_type_t](#) { [CDI_UNKNOWN](#) = 0, [CDI_NETWORK](#) = 1, [CDI_STORAGE](#) = 2 }

Funktionen

- void [cdi_init](#) (void)
- void [cdi_run_drivers](#) (void)
- void [cdi_driver_init](#) (struct [cdi_driver](#) *driver)
- void [cdi_driver_destroy](#) (struct [cdi_driver](#) *driver)
- void [cdi_driver_register](#) (struct [cdi_driver](#) *driver)

6.3.1 Makro-Dokumentation

6.3.1.1 #define CDI_STANDALONE

Definiert in Zeile 21 der Datei cdi.h.

6.3.2 Dokumentation der Aufzählungstypen

6.3.2.1 enum cdi_device_type_t

Aufzählungswerte:

```
CDI_UNKNOWN
CDI_NETWORK
CDI_STORAGE
```

Definiert in Zeile 23 der Datei cdi.h.

```
23     {
24     CDI_UNKNOWN           = 0,
25     CDI_NETWORK          = 1,
26     CDI_STORAGE          = 2
27 } cdi_device_type_t;
```

6.3.3 Dokumentation der Funktionen

6.3.3.1 void cdi_driver_destroy (struct cdi_driver * *driver*)

Deinitialisiert die Datenstrukturen fuer einen Treiber (gibt die devices-Liste frei)

6.3.3.2 void cdi_driver_init (struct cdi_driver * *driver*)

Initialisiert die Datenstrukturen fuer einen Treiber (erzeugt die devices-Liste)

6.3.3.3 void cdi_driver_register (struct cdi_driver * *driver*)

Registriert den Treiber fuer ein neues Geraet

Parameter:

driver Zu registrierender Treiber

6.3.3.4 void cdi_init (void)

Muss vor dem ersten Aufruf einer anderen CDI-Funktion aufgerufen werden. Initialisiert interne Datenstruktur der Implementierung fuer das jeweilige Betriebssystem.

Ein wiederholter Aufruf bleibt ohne Effekt.

6.3.3.5 void cdi_run_drivers (void)

Fuehrt alle registrierten Treiber aus. Nach dem Aufruf dieser Funktion duerfen keine weiteren Befehle ausgefuehrt werden, da nicht definiert ist, ob und wann die Funktion zurueckkehrt.

6.4 dma.h-Dateireferenz

```
#include <stdint.h>
#include <stdio.h>
#include "cdi.h"
```

Datenstrukturen

- struct [cdi_dma_handle](#)

Makrodefinitionen

- #define [CDI_DMA_MODE_READ](#) (1 << 2)
- #define [CDI_DMA_MODE_WRITE](#) (2 << 2)
- #define [CDI_DMA_MODE_ON_DEMAND](#) (0 << 6)
- #define [CDI_DMA_MODE_SINGLE](#) (1 << 6)
- #define [CDI_DMA_MODE_BLOCK](#) (2 << 6)

Funktionen

- int [cdi_dma_open](#) (struct [cdi_dma_handle](#) *handle, uint8_t channel, uint8_t mode, size_t length, void *buffer)
- int [cdi_dma_read](#) (struct [cdi_dma_handle](#) *handle)
- int [cdi_dma_write](#) (struct [cdi_dma_handle](#) *handle)
- int [cdi_dma_close](#) (struct [cdi_dma_handle](#) *handle)

6.4.1 Makro-Dokumentation

6.4.1.1 #define CDI_DMA_MODE_BLOCK (2 << 6)

Definiert in Zeile 34 der Datei dma.h.

6.4.1.2 #define CDI_DMA_MODE_ON_DEMAND (0 << 6)

Definiert in Zeile 32 der Datei dma.h.

6.4.1.3 #define CDI_DMA_MODE_READ (1 << 2)

Definiert in Zeile 29 der Datei dma.h.

6.4.1.4 #define CDI_DMA_MODE_SINGLE (1 << 6)

Definiert in Zeile 33 der Datei dma.h.

6.4.1.5 #define CDI_DMA_MODE_WRITE (2 << 2)

Definiert in Zeile 31 der Datei dma.h.

6.4.2 Dokumentation der Funktionen

6.4.2.1 `int cdi_dma_close (struct cdi_dma_handle * handle)`

Schliesst das DMA-Handle wieder

Rückgabe:

0 bei Erfolg, -1 im Fehlerfall

6.4.2.2 `int cdi_dma_open (struct cdi_dma_handle * handle, uint8_t channel, uint8_t mode, size_t length, void * buffer)`

Initialisiert einen Transport per DMA

Rückgabe:

0 bei Erfolg, -1 im Fehlerfall

6.4.2.3 `int cdi_dma_read (struct cdi_dma_handle * handle)`

Liest Daten per DMA ein

Rückgabe:

0 bei Erfolg, -1 im Fehlerfall

6.4.2.4 `int cdi_dma_write (struct cdi_dma_handle * handle)`

Schreibt Daten per DMA

Rückgabe:

0 bei Erfolg, -1 im Fehlerfall

6.5 fs.h-Dateireferenz

```
#include <stdio.h>
#include "cdi.h"
#include "cdi/lists.h"
#include "cdi/cache.h"
#include <sys/types.h>
```

Datenstrukturen

- struct [cdi_fs_driver](#)
- struct [cdi_fs_filesystem](#)
- struct [cdi_fs_stream](#)
- struct [cdi_fs_res_flags](#)
- struct [cdi_fs_res](#)
- struct [cdi_fs_res_res](#)
- struct [cdi_fs_res_file](#)
- struct [cdi_fs_res_dir](#)
- struct [cdi_fs_res_link](#)
- struct [cdi_fs_res_special](#)
- struct [cdi_fs_acl_entry](#)
- struct [cdi_fs_acl_entry_usr_num](#)
- struct [cdi_fs_acl_entry_usr_str](#)
- struct [cdi_fs_acl_entry_grp_num](#)
- struct [cdi_fs_acl_entry_grp_str](#)

Aufzählungen

- enum [cdi_fs_error_t](#) {
 [CDI_FS_ERROR_NONE](#) = 0, [CDI_FS_ERROR_IO](#), [CDI_FS_ERROR_ONS](#), [CDI_FS_ERROR_-](#)
 [RNF](#),
 [CDI_FS_ERROR_EOF](#), [CDI_FS_ERROR_RO](#), [CDI_FS_ERROR_INTERNAL](#), [CDI_FS_-](#)
 [ERROR_NOT_IMPLEMENTED](#),
 [CDI_FS_ERROR_UNKNOWN](#) }
- enum [cdi_fs_meta_t](#) {
 [CDI_FS_META_SIZE](#), [CDI_FS_META_USEDBLOCKS](#), [CDI_FS_META_BESTBLOCKSZ](#),
 [CDI_FS_META_BLOCKSZ](#),
 [CDI_FS_META_CREATETIME](#), [CDI_FS_META_ACCESSTIME](#), [CDI_FS_META_-](#)
 [CHANGETIME](#) }
- enum [cdi_fs_res_type_t](#) { [CDI_FS_BLOCK](#), [CDI_FS_CHAR](#), [CDI_FS_FIFO](#), [CDI_FS_SOCKET](#) }
- enum [cdi_fs_res_class_t](#) { [CDI_FS_CLASS_FILE](#), [CDI_FS_CLASS_DIR](#), [CDI_FS_CLASS_-](#)
 [LINK](#), [CDI_FS_CLASS_SPECIAL](#) }
- enum [cdi_fs_lock_t](#) { [CDI_FS_LOCK_NONE](#), [CDI_FS_LOCK_WRITE](#), [CDI_FS_LOCK_ALL](#) }
- enum [cdi_fs_acl_entry_type_t](#) { [CDI_FS_ACL_USER_NUMERIC](#), [CDI_FS_ACL_USER_-](#)
 [STRING](#), [CDI_FS_ACL_GROUP_NUMERIC](#), [CDI_FS_ACL_GROUP_STRING](#) }

Funktionen

- void `cdi_fs_driver_init` (struct `cdi_fs_driver` *driver)
- void `cdi_fs_driver_destroy` (struct `cdi_fs_driver` *driver)
- void `cdi_fs_driver_register` (struct `cdi_fs_driver` *driver)
- size_t `cdi_fs_data_read` (struct `cdi_fs_filesystem` *fs, uint64_t start, size_t size, void *buffer)
- size_t `cdi_fs_data_write` (struct `cdi_fs_filesystem` *fs, uint64_t start, size_t size, const void *buffer)

6.5.1 Dokumentation der Aufzählungstypen

6.5.1.1 enum `cdi_fs_acl_entry_type_t`

Die Berechtigungen werden mit Access controll lists, kurz ACLs verwaltet. Diese werden in Form von Listen gespeichert. Diese Listen enthalten eintraege von verschiedenen Typen.

Aufzählungswerte:

- CDI_FS_ACL_USER_NUMERIC*** Eine UID
- CDI_FS_ACL_USER_STRING*** Ein Benutzername als String
- CDI_FS_ACL_GROUP_NUMERIC*** Eine GID
- CDI_FS_ACL_GROUP_STRING*** Ein Gruppenname als String

Definiert in Zeile 563 der Datei fs.h.

```
563     {
564     CDI_FS_ACL_USER_NUMERIC,
565     CDI_FS_ACL_USER_STRING,
566     CDI_FS_ACL_GROUP_NUMERIC,
567     CDI_FS_ACL_GROUP_STRING
568     } cdi_fs_acl_entry_type_t;
```

6.5.1.2 enum `cdi_fs_error_t`

Aufzählungswerte:

- CDI_FS_ERROR_NONE*** Kein Fehler aufgetreten
- CDI_FS_ERROR_IO*** Fehler bei Eingabe/Ausgabeoperationen
- CDI_FS_ERROR_ONS*** Operation nicht unterstuetzt
- CDI_FS_ERROR_RNF*** Ressource nicht gefunden
- CDI_FS_ERROR_EOF*** Beim lesen einer Datei wurde das Ende erreicht
- CDI_FS_ERROR_RO*** Dateisystem ist nur lesbar und es wurde versucht zu schreiben
- CDI_FS_ERROR_INTERNAL*** Interner Fehler
- CDI_FS_ERROR_NOT_IMPLEMENTED*** Funktion noch nicht implementiert
- CDI_FS_ERROR_UNKNOWN*** Unbekannter Fehler

Definiert in Zeile 97 der Datei fs.h.

```

97         {
99     CDI_FS_ERROR_NONE = 0,
101     CDI_FS_ERROR_IO,
103     CDI_FS_ERROR_ONS,
105     CDI_FS_ERROR_RNF,
107     CDI_FS_ERROR_EOF,
109     CDI_FS_ERROR_RO,
111     CDI_FS_ERROR_INTERNAL,
113     CDI_FS_ERROR_NOT_IMPLEMENTED,
115     CDI_FS_ERROR_UNKNOWN
116 } cdi_fs_error_t;

```

6.5.1.3 enum cdi_fs_lock_t

Dieser Typ dient dazu Ressourcen ganz oder teilweise zu sperren

Aufzählungswerte:

CDI_FS_LOCK_NONE
CDI_FS_LOCK_WRITE
CDI_FS_LOCK_ALL

Definiert in Zeile 214 der Datei fs.h.

```

214         {
215     CDI_FS_LOCK_NONE,
216     CDI_FS_LOCK_WRITE,
217     CDI_FS_LOCK_ALL
218 } cdi_fs_lock_t;

```

6.5.1.4 enum cdi_fs_meta_t

Metaeigenschaften, die Ressourcen haben koennen

Aufzählungswerte:

CDI_FS_META_SIZE R Groesse der Datei auslesen
CDI_FS_META_USEDBLOCKS R Anzahl der Benutzten Dateisystemblocks (Irgendwo muesste man dann auch auf diese Blockgroesse zurgreifen koennen)
CDI_FS_META_BESTBLOCKSZ R Optimale Blockgroesse mit der man auf die Datei zugreifen sollte
CDI_FS_META_BLOCKSZ R Interne Blockgroesse fuer USEDBLOCKS
CDI_FS_META_CREATETIME R Zeitpunkt an dem die Ressource erstellt wurde
CDI_FS_META_ACCESSTIME RW Letzter Zugriff auf die Ressource, auch lesend
CDI_FS_META_CHANGETIME RW Letzte Veraenderung der Ressource

Definiert in Zeile 136 der Datei fs.h.

```

136         {
138     CDI_FS_META_SIZE,
141     CDI_FS_META_USEDBLOCKS,
143     CDI_FS_META_BESTBLOCKSZ,

```

```

145     CDI_FS_META_BLOCKSZ,
147     CDI_FS_META_CREATETIME,
149     CDI_FS_META_ACCESSTIME,
151     CDI_FS_META_CHANGETIME
152 } cdi_fs_meta_t;

```

6.5.1.5 enum cdi_fs_res_class_t

Konstanten fuer die einzelnen Klassen, um sie beim Funktionsaufruf zum zuweisen einer Klasse, identifizieren zu koennen.

Aufzählungswerte:

```

CDI_FS_CLASS_FILE
CDI_FS_CLASS_DIR
CDI_FS_CLASS_LINK
CDI_FS_CLASS_SPECIAL

```

Definiert in Zeile 204 der Datei fs.h.

```

204     {
205     CDI_FS_CLASS_FILE,
206     CDI_FS_CLASS_DIR,
207     CDI_FS_CLASS_LINK,
208     CDI_FS_CLASS_SPECIAL
209 } cdi_fs_res_class_t;

```

6.5.1.6 enum cdi_fs_res_type_t

Typ der eine Ressource, die zu der Klasse der Spezialdateien gehoert noch genauer beschreibt

Aufzählungswerte:

```

CDI_FS_BLOCK
CDI_FS_CHAR
CDI_FS_FIFO
CDI_FS_SOCKET

```

Definiert in Zeile 193 der Datei fs.h.

```

193     {
194     CDI_FS_BLOCK,
195     CDI_FS_CHAR,
196     CDI_FS_FIFO,
197     CDI_FS_SOCKET
198 } cdi_fs_res_type_t;

```

6.5.2 Dokumentation der Funktionen

6.5.2.1 size_t cdi_fs_data_read (struct cdi_fs_filesystem * fs, uint64_t start, size_t size, void * buffer)

Quelldateien fuer ein Dateisystem lesen XXX Brauchen wir hier auch noch irgendwas erno-Maessiges?

Parameter:

fs Pointer auf die FS-Struktur des Dateisystems
start Position von der an gelesen werden soll
size Groesse des zu lesenden Datenblocks
buffer Puffer in dem die Daten abgelegt werden sollen

Rückgabe:

die Anzahl der gelesenen Bytes

6.5.2.2 `size_t cdi_fs_data_write (struct cdi_fs_filesystem *fs, uint64_t start, size_t size, const void *buffer)`

Quellmedium eines Dateisystems beschreiben XXX Brauchen wir hier auch noch irgendwas erno-Maessiges?

Parameter:

fs Pointer auf die FS-Struktur des Dateisystems
start Position an die geschrieben werden soll
size Groesse des zu schreibenden Datenblocks
buffer Puffer aus dem die Daten gelesen werden sollen

Rückgabe:

die Anzahl der geschriebenen Bytes

6.5.2.3 `void cdi_fs_driver_destroy (struct cdi_fs_driver *driver)`

Dateisystemtreiber-Struktur zerstören

Parameter:

driver Zeiger auf den Treiber

6.5.2.4 `void cdi_fs_driver_init (struct cdi_fs_driver *driver)`

Dateisystemtreiber-Struktur initialisieren

Parameter:

driver Zeiger auf den Treiber

6.5.2.5 `void cdi_fs_driver_register (struct cdi_fs_driver *driver)`

Dateisystemtreiber registrieren

Parameter:

driver Zeiger auf den Treiber

6.6 io.h-Dateireferenz

```
#include <stdint.h>
```

6.7 lists.h-Dateireferenz

```
#include <stddef.h>
#include <stdint.h>
```

Typdefinitionen

- typedef struct cdi_list_implementation * [cdi_list_t](#)

Funktionen

- [cdi_list_t cdi_list_create](#) (void)
- void [cdi_list_destroy](#) ([cdi_list_t](#) list)
- [cdi_list_t cdi_list_push](#) ([cdi_list_t](#) list, void *value)
- void * [cdi_list_pop](#) ([cdi_list_t](#) list)
- size_t [cdi_list_empty](#) ([cdi_list_t](#) list)
- void * [cdi_list_get](#) ([cdi_list_t](#) list, size_t index)
- [cdi_list_t cdi_list_insert](#) ([cdi_list_t](#) list, size_t index, void *value)
- void * [cdi_list_remove](#) ([cdi_list_t](#) list, size_t index)
- size_t [cdi_list_size](#) ([cdi_list_t](#) list)

6.7.1 Dokumentation der benutzerdefinierten Typen

6.7.1.1 typedef struct cdi_list_implementation* cdi_list_t

Repraesentiert eine Liste.

Der Felder der Struktur sind implementierungsabhaengig. Zum Zugriff auf Listen muessen immer die spezifizierten Funktionen benutzt werden.

Definiert in Zeile 22 der Datei lists.h.

6.7.2 Dokumentation der Funktionen

6.7.2.1 cdi_list_t cdi_list_create (void)

Erzeugt eine neue Liste

Rückgabe:

Neu erzeugte Liste oder NULL, falls kein Speicher reserviert werden konnte

6.7.2.2 void cdi_list_destroy (cdi_list_t list)

Gibt eine Liste frei (Werte der Listenglieder müssen bereits freigegeben sein)

6.7.2.3 size_t cdi_list_empty (cdi_list_t list)

Prueft, ob die Liste leer ist.

Parameter:

list Liste, die ueberprueft werden soll

Rückgabe:

1, wenn die Liste leer ist; 0, wenn sie Elemente enthaelt

6.7.2.4 void* cdi_list_get (cdi_list_t list, size_t index)

Gibt ein Listenelement zurueck

Parameter:

list Liste, aus der das Element gelesen werden soll

index Index des zurueckzugebenden Elements

Rückgabe:

Das angeforderte Element oder NULL, wenn kein Element mit dem angegebenen Index existiert.

6.7.2.5 cdi_list_t cdi_list_insert (cdi_list_t list, size_t index, void * value)

Fuegt ein neues Listenelement ein. Der Index aller Elemente, die bisher einen groesseeren oder gleich grossen Index haben, verschieben sich um eine Position nach hinten.

Parameter:

list Liste, in die eingefuegt werden soll

index Zukuenftiger Index des neu einzufuegenden Elements

value Neu einzufuegendes Element

Rückgabe:

Die Liste, in die eingefuegt wurde, oder NULL, wenn nicht eingefuegt werden konnte (z.B. weil der Index zu gross ist)

6.7.2.6 void* cdi_list_pop (cdi_list_t list)

Entfernt ein Element am Anfang (Index 0) der Liste und gibt seinen Wert zurueck.

Parameter:

list Liste, aus der das Element entnommen werden soll

Rückgabe:

Das entfernte Element oder NULL, wenn die Liste leer war

6.7.2.7 cdi_list_t cdi_list_push (cdi_list_t list, void * value)

Fuegt ein neues Element am Anfang (Index 0) der Liste ein

Parameter:

list Liste, in die eingefuegt werden soll

value Einzufuegendes Element

Rückgabe:

Die Liste, in die eingefuegt wurde, oder NULL, wenn das Element nicht eingefuegt werden konnte (z.B. kein Speicher frei).

6.7.2.8 void* cdi_list_remove (cdi_list_t list, size_t index)

Loescht ein Listenelement

Parameter:

list Liste, aus der entfernt werden soll

index Index des zu entfernenden Elements

Rückgabe:

Das entfernte Element oder NULL, wenn kein Element mit dem angegebenen Index existiert.

6.7.2.9 size_t cdi_list_size (cdi_list_t list)

Gibt die Laenge der Liste zurueck

Parameter:

list Liste, deren Laenge zurueckgegeben werden soll

6.8 misc.h-Dateireferenz

```
#include <stdint.h>
#include "cdi.h"
```

Funktionen

- void `cdi_register_irq` (uint8_t irq, void(*handler)(struct `cdi_device` *), struct `cdi_device` *device)
- int `cdi_alloc_phys_mem` (size_t size, void **vaddr, void **paddr)
- void * `cdi_alloc_phys_addr` (size_t size, uintptr_t paddr)
- int `cdi_ioports_alloc` (uint16_t start, uint16_t count)
- int `cdi_ioports_free` (uint16_t start, uint16_t count)
- void `cdi_sleep_ms` (uint32_t ms)

6.8.1 Dokumentation der Funktionen

6.8.1.1 void* `cdi_alloc_phys_addr` (size_t *size*, uintptr_t *paddr*)

Reserviert physisch zusammenhaengenden Speicher an einer definierten Adresse.

Parameter:

- size* Groesse des benoetigten Speichers in Bytes
- paddr* Physikalische Adresse des angeforderten Speicherbereichs

Rückgabe:

Virtuelle Adresse, wenn Speicher reserviert wurde, sonst 0

6.8.1.2 int `cdi_alloc_phys_mem` (size_t *size*, void ** *vaddr*, void ** *paddr*)

Reserviert physisch zusammenhaengenden Speicher.

Parameter:

- size* Groesse des benoetigten Speichers in Bytes
- vaddr* Pointer, in den die virtuelle Adresse des reservierten Speichers geschrieben wird.
- paddr* Pointer, in den die physische Adresse des reservierten Speichers geschrieben wird.

Rückgabe:

0 wenn der Speicher erfolgreich reserviert wurde, -1 sonst

6.8.1.3 int `cdi_ioports_alloc` (uint16_t *start*, uint16_t *count*)

Reserviert IO-Ports

Rückgabe:

0 wenn die Ports erfolgreich reserviert wurden, -1 sonst.

6.8.1.4 int cdi_ioports_free (uint16_t start, uint16_t count)

Gibt reservierte IO-Ports frei

Rückgabe:

0 wenn die Ports erfolgreich freigegeben wurden, -1 sonst.

6.8.1.5 void cdi_register_irq (uint8_t irq, void(*)(struct cdi_device *) handler, struct cdi_device * device)

Registriert einen neuen IRQ-Handler.

Parameter:

irq Nummer des zu reservierenden IRQ

handler Handlerfunktion

device Geraet, das dem Handler als Parameter uebergeben werden soll

6.8.1.6 void cdi_sleep_ms (uint32_t ms)

Unterbricht die Ausfuehrung fuer mehrere Millisekunden

6.9 net.h-Dateireferenz

```
#include <stdint.h>
#include <stddef.h>
#include "cdi.h"
```

Datenstrukturen

- struct [cdi_net_device](#)
- struct [cdi_net_driver](#)

Funktionen

- void [cdi_net_driver_init](#) (struct [cdi_net_driver](#) *driver)
- void [cdi_net_driver_destroy](#) (struct [cdi_net_driver](#) *driver)
- void [cdi_net_device_init](#) (struct [cdi_net_device](#) *device)
- void [cdi_net_receive](#) (struct [cdi_net_device](#) *device, void *buffer, size_t size)

6.9.1 Dokumentation der Funktionen

6.9.1.1 void [cdi_net_device_init](#) (struct [cdi_net_device](#) * *device*)

Initialisiert eine neue Netzwerkkarte

6.9.1.2 void [cdi_net_driver_destroy](#) (struct [cdi_net_driver](#) * *driver*)

Deinitialisiert die Datenstrukturen fuer einen Netzwerktreiber (gibt die devices-Liste frei)

6.9.1.3 void [cdi_net_driver_init](#) (struct [cdi_net_driver](#) * *driver*)

Initialisiert die Datenstrukturen fuer einen Netzkerktreiber (erzeugt die devices-Liste)

6.9.1.4 void [cdi_net_receive](#) (struct [cdi_net_device](#) * *device*, void * *buffer*, size_t *size*)

Wird von Netzwerktreibern aufgerufen, wenn ein Netzwerkpaket empfangen wurde.

6.10 pci.h-Dateireferenz

```
#include <stdint.h>
#include "cdi.h"
#include "cdi/lists.h"
```

Datenstrukturen

- struct [cdi_pci_device](#)
- struct [cdi_pci_resource](#)

Aufzählungen

- enum [cdi_res_t](#) { [CDI_PCI_MEMORY](#), [CDI_PCI_IOPORTS](#) }

Funktionen

- void [cdi_pci_get_all_devices](#) ([cdi_list_t](#) list)
- void [cdi_pci_device_destroy](#) (struct [cdi_pci_device](#) *device)
- void [cdi_pci_alloc_ioports](#) (struct [cdi_pci_device](#) *device)
- void [cdi_pci_free_ioports](#) (struct [cdi_pci_device](#) *device)
- void [cdi_pci_alloc_memory](#) (struct [cdi_pci_device](#) *device)
- void [cdi_pci_free_memory](#) (struct [cdi_pci_device](#) *device)

6.10.1 Dokumentation der Aufzählungstypen

6.10.1.1 enum [cdi_res_t](#)

Aufzählungswerte:

CDI_PCI_MEMORY
CDI_PCI_IOPORTS

Definiert in Zeile 34 der Datei pci.h.

```
34     {
35     CDI_PCI_MEMORY,
36     CDI_PCI_IOPORTS
37 } cdi_res_t;
```

6.10.2 Dokumentation der Funktionen

6.10.2.1 void [cdi_pci_alloc_ioports](#) (struct [cdi_pci_device](#) * *device*)

Reserviert die IO-Ports des PCI-Geraets fuer den Treiber

6.10.2.2 void [cdi_pci_alloc_memory](#) (struct [cdi_pci_device](#) * *device*)

Reserviert den MMIO-Speicher des PCI-Geraets fuer den Treiber

6.10.2.3 void cdi_pci_device_destroy (struct cdi_pci_device * device)

Gibt die Information zu einem PCI-Geraet frei

6.10.2.4 void cdi_pci_free_ioports (struct cdi_pci_device * device)

Gibt die IO-Ports des PCI-Geraets frei

6.10.2.5 void cdi_pci_free_memory (struct cdi_pci_device * device)

Gibt den MMIO-Speicher des PCI-Geraets frei

6.10.2.6 void cdi_pci_get_all_devices (cdi_list_t list)

Gibt alle PCI-Geraete im System zurueck. Die Geraete werden dazu in die uebergebene Liste eingefuegt.

6.11 scsi.h-Dateireferenz

```
#include <stdint.h>
#include "cdi.h"
```

Datenstrukturen

- struct [cdi_scsi_device](#)
- struct [cdi_scsi_driver](#)

Funktionen

- void [cdi_scsi_driver_init](#) (struct [cdi_scsi_driver](#) *driver)
- void [cdi_scsi_driver_destroy](#) (struct [cdi_scsi_driver](#) *driver)
- void [cdi_scsi_driver_register](#) (struct [cdi_scsi_driver](#) *driver)

6.11.1 Dokumentation der Funktionen

6.11.1.1 void [cdi_scsi_driver_destroy](#) (struct [cdi_scsi_driver](#) * *driver*)

Deinitialisiert die Datenstrukturen fuer einen SCSI-Treiber (gibt die devices-Liste frei)

6.11.1.2 void [cdi_scsi_driver_init](#) (struct [cdi_scsi_driver](#) * *driver*)

Initialisiert die Datenstrukturen fuer einen SCSI-Treiber (erzeugt die devices-Liste)

6.11.1.3 void [cdi_scsi_driver_register](#) (struct [cdi_scsi_driver](#) * *driver*)

Registriert einen SCSI-treiber

6.12 storage.h-Dateireferenz

```
#include <stdint.h>
#include "cdi.h"
```

Datenstrukturen

- struct [cdi_storage_device](#)
- struct [cdi_storage_driver](#)

Funktionen

- void [cdi_storage_driver_init](#) (struct [cdi_storage_driver](#) *driver)
- void [cdi_storage_driver_destroy](#) (struct [cdi_storage_driver](#) *driver)
- void [cdi_storage_driver_register](#) (struct [cdi_storage_driver](#) *driver)

6.12.1 Dokumentation der Funktionen

6.12.1.1 void [cdi_storage_driver_destroy](#) (struct [cdi_storage_driver](#) * *driver*)

Deinitialisiert die Datenstrukturen fuer einen Massenspeichertreiber (gibt die devices-Liste frei)

6.12.1.2 void [cdi_storage_driver_init](#) (struct [cdi_storage_driver](#) * *driver*)

Initialisiert die Datenstrukturen fuer einen Massenspeichertreiber (erzeugt die devices-Liste)

6.12.1.3 void [cdi_storage_driver_register](#) (struct [cdi_storage_driver](#) * *driver*)

Registriert einen Massenspeichertreiber

Index

- acl
 - cdi_fs_res, 31
- address
 - cdi_pci_resource, 49
- assign_class
 - cdi_fs_res_res, 40
- ax
 - cdi_bios_registers, 15
- bios.h, 55
 - cdi_bios_int10, 55
- block_count
 - cdi_storage_device, 52
- block_size
 - cdi_cache, 17
 - cdi_storage_device, 52
- browse
 - cdi_fs_res_flags, 37
- buffer
 - cdi_dma_handle, 20
- bus
 - cdi_pci_device, 47
- bx
 - cdi_bios_registers, 15
- cache.h, 56
 - cdi_cache_block_dirty, 56
 - cdi_cache_block_get, 57
 - cdi_cache_block_release, 57
 - cdi_cache_create, 57
 - cdi_cache_destroy, 57
 - cdi_cache_read_block_t, 56
 - cdi_cache_sync, 58
 - cdi_cache_write_block_t, 56
- cdi.h, 59
 - CDI_NETWORK, 59
 - CDI_STORAGE, 59
 - CDI_UNKNOWN, 59
 - cdi_device_type_t, 59
 - cdi_driver_destroy, 60
 - cdi_driver_init, 60
 - cdi_driver_register, 60
 - cdi_init, 60
 - cdi_run_drivers, 60
 - CDI_STANDALONE, 59
 - CDI_FS_ACL_GROUP_NUMERIC
 - fs.h, 64
 - CDI_FS_ACL_GROUP_STRING
 - fs.h, 64
 - CDI_FS_ACL_USER_NUMERIC
 - fs.h, 64
 - CDI_FS_ACL_USER_STRING
 - fs.h, 64
 - CDI_FS_BLOCK
 - fs.h, 66
 - CDI_FS_CHAR
 - fs.h, 66
 - CDI_FS_CLASS_DIR
 - fs.h, 66
 - CDI_FS_CLASS_FILE
 - fs.h, 66
 - CDI_FS_CLASS_LINK
 - fs.h, 66
 - CDI_FS_CLASS_SPECIAL
 - fs.h, 66
 - CDI_FS_ERROR_EOF
 - fs.h, 64
 - CDI_FS_ERROR_INTERNAL
 - fs.h, 64
 - CDI_FS_ERROR_IO
 - fs.h, 64
 - CDI_FS_ERROR_NONE
 - fs.h, 64
 - CDI_FS_ERROR_NOT_IMPLEMENTED
 - fs.h, 64
 - CDI_FS_ERROR_ONS
 - fs.h, 64
 - CDI_FS_ERROR_RNF
 - fs.h, 64
 - CDI_FS_ERROR_RO
 - fs.h, 64
 - CDI_FS_ERROR_UNKNOWN
 - fs.h, 64
 - CDI_FS_FIFO
 - fs.h, 66
 - CDI_FS_LOCK_ALL
 - fs.h, 65
 - CDI_FS_LOCK_NONE
 - fs.h, 65
 - CDI_FS_LOCK_WRITE

- fs.h, 65
- CDI_FS_META_ACCESSTIME
 - fs.h, 65
- CDI_FS_META_BESTBLOCKSZ
 - fs.h, 65
- CDI_FS_META_BLOCKSZ
 - fs.h, 65
- CDI_FS_META_CHANGETIME
 - fs.h, 65
- CDI_FS_META_CREATETIME
 - fs.h, 65
- CDI_FS_META_SIZE
 - fs.h, 65
- CDI_FS_META_USEDBLOCKS
 - fs.h, 65
- CDI_FS_SOCKET
 - fs.h, 66
- CDI_NETWORK
 - cdi.h, 59
- CDI_PCI_IOPORTS
 - pci.h, 75
- CDI_PCI_MEMORY
 - pci.h, 75
- CDI_STORAGE
 - cdi.h, 59
- CDI_UNKNOWN
 - cdi.h, 59
- cdi_alloc_phys_addr
 - misc.h, 72
- cdi_alloc_phys_mem
 - misc.h, 72
- cdi_bios_int10
 - bios.h, 55
- cdi_bios_memory, 13
 - dest, 13
 - size, 13
 - src, 13
- cdi_bios_registers, 15
 - ax, 15
 - bx, 15
 - cx, 15
 - di, 15
 - ds, 15
 - dx, 15
 - es, 16
 - si, 15
- cdi_cache, 17
 - block_size, 17
- cdi_cache_block, 18
 - data, 18
 - number, 18
 - private, 18
- cdi_cache_block_dirty
 - cache.h, 56
- cdi_cache_block_get
 - cache.h, 57
- cdi_cache_block_release
 - cache.h, 57
- cdi_cache_create
 - cache.h, 57
- cdi_cache_destroy
 - cache.h, 57
- cdi_cache_read_block_t
 - cache.h, 56
- cdi_cache_sync
 - cache.h, 58
- cdi_cache_write_block_t
 - cache.h, 56
- cdi_device, 19
 - driver, 19
 - name, 19
 - type, 19
- cdi_device_type_t
 - cdi.h, 59
- cdi_dma_close
 - dma.h, 62
- cdi_dma_handle, 20
 - buffer, 20
 - channel, 20
 - file, 20
 - length, 20
 - mode, 20
- CDI_DMA_MODE_BLOCK
 - dma.h, 61
- CDI_DMA_MODE_ON_DEMAND
 - dma.h, 61
- CDI_DMA_MODE_READ
 - dma.h, 61
- CDI_DMA_MODE_SINGLE
 - dma.h, 61
- CDI_DMA_MODE_WRITE
 - dma.h, 61
- cdi_dma_open
 - dma.h, 62
- cdi_dma_read
 - dma.h, 62
- cdi_dma_write
 - dma.h, 62
- cdi_driver, 21
 - destroy, 21
 - devices, 21
 - init_device, 21
 - name, 21
 - remove_device, 21
 - type, 21
- cdi_driver_destroy
 - cdi.h, 60
- cdi_driver_init

- cdi.h, 60
- cdi_driver_register
 - cdi.h, 60
- cdi_fs_acl_entry, 22
 - flags, 22
 - type, 22
- cdi_fs_acl_entry_grp_num, 23
 - entry, 23
 - group_id, 23
- cdi_fs_acl_entry_grp_str, 24
 - entry, 24
 - group_name, 24
- cdi_fs_acl_entry_type_t
 - fs.h, 64
- cdi_fs_acl_entry_usr_num, 25
 - entry, 25
 - user_id, 25
- cdi_fs_acl_entry_usr_str, 26
 - entry, 26
 - user_name, 26
- cdi_fs_data_read
 - fs.h, 66
- cdi_fs_data_write
 - fs.h, 67
- cdi_fs_driver, 27
 - drv, 27
 - fs_destroy, 27
 - fs_init, 27
- cdi_fs_driver_destroy
 - fs.h, 67
- cdi_fs_driver_init
 - fs.h, 67
- cdi_fs_driver_register
 - fs.h, 67
- cdi_fs_error_t
 - fs.h, 64
- cdi_fs_filesystem, 28
 - device, 28
 - driver, 28
 - error, 28
 - opaque, 28
 - read_only, 28
 - root_res, 28
- cdi_fs_lock_t
 - fs.h, 65
- cdi_fs_meta_t
 - fs.h, 65
- cdi_fs_res, 30
 - acl, 31
 - children, 31
 - dir, 31
 - file, 31
 - flags, 31
 - link, 32
 - link_path, 31
 - loaded, 30
 - lock, 30
 - name, 30
 - parent, 31
 - res, 31
 - special, 32
 - stream_cnt, 30
 - type, 32
- cdi_fs_res_class_t
 - fs.h, 66
- cdi_fs_res_dir, 33
 - create_child, 33
 - list, 33
- cdi_fs_res_file, 34
 - executable, 34
 - read, 34
 - truncate, 35
 - write, 34
- cdi_fs_res_flags, 36
 - browse, 37
 - create_child, 37
 - execute, 37
 - move, 36
 - read, 36
 - read_link, 37
 - remove, 36
 - rename, 36
 - write, 36
 - write_link, 37
- cdi_fs_res_link, 38
 - read_link, 38
 - write_link, 38
- cdi_fs_res_res, 39
 - assign_class, 40
 - load, 39
 - meta_read, 41
 - meta_write, 41
 - move, 40
 - remove, 40
 - remove_class, 41
 - rename, 40
 - unload, 39
- cdi_fs_res_special, 43
 - dev_read, 43
 - dev_write, 43
- cdi_fs_res_type_t
 - fs.h, 66
- cdi_fs_stream, 44
 - error, 44
 - fs, 44
 - res, 44
- cdi_init
 - cdi.h, 60

- cdi_ioports_alloc
 - misc.h, 72
- cdi_ioports_free
 - misc.h, 72
- cdi_list_create
 - lists.h, 69
- cdi_list_destroy
 - lists.h, 69
- cdi_list_empty
 - lists.h, 69
- cdi_list_get
 - lists.h, 70
- cdi_list_insert
 - lists.h, 70
- cdi_list_pop
 - lists.h, 70
- cdi_list_push
 - lists.h, 70
- cdi_list_remove
 - lists.h, 71
- cdi_list_size
 - lists.h, 71
- cdi_list_t
 - lists.h, 69
- cdi_net_device, 45
 - dev, 45
 - ip, 45
 - mac, 45
 - number, 45
 - send_packet, 45
- cdi_net_device_init
 - net.h, 74
- cdi_net_driver, 46
 - drv, 46
- cdi_net_driver_destroy
 - net.h, 74
- cdi_net_driver_init
 - net.h, 74
- cdi_net_receive
 - net.h, 74
- cdi_pci_alloc_ioports
 - pci.h, 75
- cdi_pci_alloc_memory
 - pci.h, 75
- cdi_pci_device, 47
 - bus, 47
 - class_id, 47
 - dev, 47
 - device_id, 47
 - function, 47
 - irq, 48
 - resources, 48
 - rev_id, 47
 - vendor_id, 47
- cdi_pci_device_destroy
 - pci.h, 75
- cdi_pci_free_ioports
 - pci.h, 76
- cdi_pci_free_memory
 - pci.h, 76
- cdi_pci_get_all_devices
 - pci.h, 76
- cdi_pci_resource, 49
 - address, 49
 - index, 49
 - length, 49
 - start, 49
 - type, 49
- cdi_register_irq
 - misc.h, 73
- cdi_res_t
 - pci.h, 75
- cdi_run_drivers
 - cdi.h, 60
- cdi_scsi_device, 50
 - dev, 50
- cdi_scsi_driver, 51
 - drv, 51
- cdi_scsi_driver_destroy
 - scsi.h, 77
- cdi_scsi_driver_init
 - scsi.h, 77
- cdi_scsi_driver_register
 - scsi.h, 77
- cdi_sleep_ms
 - misc.h, 73
- CDI_STANDALONE
 - cdi.h, 59
- cdi_storage_device, 52
 - block_count, 52
 - block_size, 52
 - dev, 52
- cdi_storage_driver, 53
 - drv, 53
 - read_blocks, 53
 - write_blocks, 53
- cdi_storage_driver_destroy
 - storage.h, 78
- cdi_storage_driver_init
 - storage.h, 78
- cdi_storage_driver_register
 - storage.h, 78
- channel
 - cdi_dma_handle, 20
- children
 - cdi_fs_res, 31
- class_id
 - cdi_pci_device, 47

- Core, 11
- create_child
 - cdi_fs_res_dir, 33
 - cdi_fs_res_flags, 37
- cx
 - cdi_bios_registers, 15
- data
 - cdi_cache_block, 18
- dest
 - cdi_bios_memory, 13
- destroy
 - cdi_driver, 21
- dev
 - cdi_net_device, 45
 - cdi_pci_device, 47
 - cdi_scsi_device, 50
 - cdi_storage_device, 52
- dev_read
 - cdi_fs_res_special, 43
- dev_write
 - cdi_fs_res_special, 43
- device
 - cdi_fs_filesystem, 28
- device_id
 - cdi_pci_device, 47
- devices
 - cdi_driver, 21
- di
 - cdi_bios_registers, 15
- dir
 - cdi_fs_res, 31
- dma.h, 61
 - cdi_dma_close, 62
 - CDI_DMA_MODE_BLOCK, 61
 - CDI_DMA_MODE_ON_DEMAND, 61
 - CDI_DMA_MODE_READ, 61
 - CDI_DMA_MODE_SINGLE, 61
 - CDI_DMA_MODE_WRITE, 61
 - cdi_dma_open, 62
 - cdi_dma_read, 62
 - cdi_dma_write, 62
- driver
 - cdi_device, 19
 - cdi_fs_filesystem, 28
- drv
 - cdi_fs_driver, 27
 - cdi_net_driver, 46
 - cdi_scsi_driver, 51
 - cdi_storage_driver, 53
- ds
 - cdi_bios_registers, 15
- dx
 - cdi_bios_registers, 15
- entry
 - cdi_fs_acl_entry_grp_num, 23
 - cdi_fs_acl_entry_grp_str, 24
 - cdi_fs_acl_entry_usr_num, 25
 - cdi_fs_acl_entry_usr_str, 26
- error
 - cdi_fs_filesystem, 28
 - cdi_fs_stream, 44
- es
 - cdi_bios_registers, 16
- executable
 - cdi_fs_res_file, 34
- execute
 - cdi_fs_res_flags, 37
- file
 - cdi_dma_handle, 20
 - cdi_fs_res, 31
- flags
 - cdi_fs_acl_entry, 22
 - cdi_fs_res, 31
- Fs, 7
- fs
 - cdi_fs_stream, 44
- fs.h, 63
 - CDI_FS_ACL_GROUP_NUMERIC, 64
 - CDI_FS_ACL_GROUP_STRING, 64
 - CDI_FS_ACL_USER_NUMERIC, 64
 - CDI_FS_ACL_USER_STRING, 64
 - CDI_FS_BLOCK, 66
 - CDI_FS_CHAR, 66
 - CDI_FS_CLASS_DIR, 66
 - CDI_FS_CLASS_FILE, 66
 - CDI_FS_CLASS_LINK, 66
 - CDI_FS_CLASS_SPECIAL, 66
 - CDI_FS_ERROR_EOF, 64
 - CDI_FS_ERROR_INTERNAL, 64
 - CDI_FS_ERROR_IO, 64
 - CDI_FS_ERROR_NONE, 64
 - CDI_FS_ERROR_NOT_IMPLEMENTED, 64
 - CDI_FS_ERROR_ONS, 64
 - CDI_FS_ERROR_RNF, 64
 - CDI_FS_ERROR_RO, 64
 - CDI_FS_ERROR_UNKNOWN, 64
 - CDI_FS_FIFO, 66
 - CDI_FS_LOCK_ALL, 65
 - CDI_FS_LOCK_NONE, 65
 - CDI_FS_LOCK_WRITE, 65
 - CDI_FS_META_ACCESSTIME, 65
 - CDI_FS_META_BESTBLOCKSZ, 65
 - CDI_FS_META_BLOCKSZ, 65
 - CDI_FS_META_CHANGETIME, 65
 - CDI_FS_META_CREATETIME, 65

- CDI_FS_META_SIZE, 65
- CDI_FS_META_USED_BLOCKS, 65
- CDI_FS_SOCKET, 66
- cdi_fs_acl_entry_type_t, 64
- cdi_fs_data_read, 66
- cdi_fs_data_write, 67
- cdi_fs_driver_destroy, 67
- cdi_fs_driver_init, 67
- cdi_fs_driver_register, 67
- cdi_fs_error_t, 64
- cdi_fs_lock_t, 65
- cdi_fs_meta_t, 65
- cdi_fs_res_class_t, 66
- cdi_fs_res_type_t, 66
- fs_destroy
 - cdi_fs_driver, 27
- fs_init
 - cdi_fs_driver, 27
- function
 - cdi_pci_device, 47
- group_id
 - cdi_fs_acl_entry_grp_num, 23
- group_name
 - cdi_fs_acl_entry_grp_str, 24
- index
 - cdi_pci_resource, 49
- init_device
 - cdi_driver, 21
- io.h, 68
- ip
 - cdi_net_device, 45
- irq
 - cdi_pci_device, 48
- length
 - cdi_dma_handle, 20
 - cdi_pci_resource, 49
- link
 - cdi_fs_res, 32
- link_path
 - cdi_fs_res, 31
- list
 - cdi_fs_res_dir, 33
- lists.h, 69
 - cdi_list_create, 69
 - cdi_list_destroy, 69
 - cdi_list_empty, 69
 - cdi_list_get, 70
 - cdi_list_insert, 70
 - cdi_list_pop, 70
 - cdi_list_push, 70
 - cdi_list_remove, 71
 - cdi_list_size, 71
 - cdi_list_t, 69
- load
 - cdi_fs_res_res, 39
- loaded
 - cdi_fs_res, 30
- lock
 - cdi_fs_res, 30
- mac
 - cdi_net_device, 45
- meta_read
 - cdi_fs_res_res, 41
- meta_write
 - cdi_fs_res_res, 41
- misc.h, 72
 - cdi_alloc_phys_addr, 72
 - cdi_alloc_phys_mem, 72
 - cdi_ioports_alloc, 72
 - cdi_ioports_free, 72
 - cdi_register_irq, 73
 - cdi_sleep_ms, 73
- mode
 - cdi_dma_handle, 20
- move
 - cdi_fs_res_flags, 36
 - cdi_fs_res_res, 40
- name
 - cdi_device, 19
 - cdi_driver, 21
 - cdi_fs_res, 30
- Net, 8
- net.h, 74
 - cdi_net_device_init, 74
 - cdi_net_driver_destroy, 74
 - cdi_net_driver_init, 74
 - cdi_net_receive, 74
- number
 - cdi_cache_block, 18
 - cdi_net_device, 45
- opaque
 - cdi_fs_filesystem, 28
- parent
 - cdi_fs_res, 31
- pci.h, 75
 - CDI_PCI_IOPORTS, 75
 - CDI_PCI_MEMORY, 75
 - cdi_pci_alloc_ioports, 75
 - cdi_pci_alloc_memory, 75
 - cdi_pci_device_destroy, 75
 - cdi_pci_free_ioports, 76

- cdi_pci_free_memory, 76
- cdi_pci_get_all_devices, 76
- cdi_res_t, 75
- private
 - cdi_cache_block, 18
- read
 - cdi_fs_res_file, 34
 - cdi_fs_res_flags, 36
- read_blocks
 - cdi_storage_driver, 53
- read_link
 - cdi_fs_res_flags, 37
 - cdi_fs_res_link, 38
- read_only
 - cdi_fs_filesystem, 28
- remove
 - cdi_fs_res_flags, 36
 - cdi_fs_res_res, 40
- remove_class
 - cdi_fs_res_res, 41
- remove_device
 - cdi_driver, 21
- rename
 - cdi_fs_res_flags, 36
 - cdi_fs_res_res, 40
- res
 - cdi_fs_res, 31
 - cdi_fs_stream, 44
- resources
 - cdi_pci_device, 48
- rev_id
 - cdi_pci_device, 47
- root_res
 - cdi_fs_filesystem, 28
- Scsi, 9
- scsi.h, 77
 - cdi_scsi_driver_destroy, 77
 - cdi_scsi_driver_init, 77
 - cdi_scsi_driver_register, 77
- send_packet
 - cdi_net_device, 45
- si
 - cdi_bios_registers, 15
- size
 - cdi_bios_memory, 13
- special
 - cdi_fs_res, 32
- src
 - cdi_bios_memory, 13
- start
 - cdi_pci_resource, 49
- Storage, 10
 - storage.h, 78
 - cdi_storage_driver_destroy, 78
 - cdi_storage_driver_init, 78
 - cdi_storage_driver_register, 78
 - stream_cnt
 - cdi_fs_res, 30
 - truncate
 - cdi_fs_res_file, 35
 - type
 - cdi_device, 19
 - cdi_driver, 21
 - cdi_fs_acl_entry, 22
 - cdi_fs_res, 32
 - cdi_pci_resource, 49
 - unload
 - cdi_fs_res_res, 39
 - user_id
 - cdi_fs_acl_entry_usr_num, 25
 - user_name
 - cdi_fs_acl_entry_usr_str, 26
 - vendor_id
 - cdi_pci_device, 47
 - write
 - cdi_fs_res_file, 34
 - cdi_fs_res_flags, 36
 - write_blocks
 - cdi_storage_driver, 53
 - write_link
 - cdi_fs_res_flags, 37
 - cdi_fs_res_link, 38