NAME

cd - SCSI CD-ROM driver

SYNOPSIS

device cd

DESCRIPTION

The **cd** driver provides support for a SCSI CD-ROM (Compact Disc-Read Only Memory) drive. In an attempt to look like a regular disk, the **cd** driver synthesizes a partition table, with one partition covering the entire CD-ROM. It is possible to modify this partition table using disklabel(8), but it will only last until the CD-ROM is unmounted. In general the interfaces are similar to those described by ada(4) and da(4).

As the SCSI adapter is probed during boot, the SCSI bus is scanned for devices. Any devices found which answer as CDROM (type 5) or WORM (type 4) type devices will be ‘attached’ to the **cd** driver. Prior to FreeBSD 2.1, the first device found will be attached as cd0 the next, cd1, etc. Beginning in FreeBSD 2.1 it is possible to specify what cd unit a device should come on line as; refer to scsi(4) for details on kernel configuration.

The system utility disklabel(8) may be used to read the synthesized disk label structure, which will contain correct figures for the size of the CD-ROM should that information be required.

KERNEL CONFIGURATION

Any number of CD-ROM devices may be attached to the system regardless of system configuration as all resources are dynamically allocated.

IOCTLS

The following ioctl(2) calls which apply to SCSI CD-ROM drives are defined in the header files <*sys/cdio.h*> and <*sys/disklabel.h*>.

CDIOCPLAYTRACKS (struct ioc_play_track) Start audio playback given a track address and length. The structure is defined as follows:

```
struct ioc_play_track
{
        u_char  start_track;
        u_char  start_index;
        u_char  end_track;
        u_char  end_index;
};
```

CDIOCPLAYBLOCKS (struct ioc_play_blocks) Start audio playback given a block address and length. The structure is defined as follows:

```
struct ioc_play_blocks
{
        int     blk;
        int     len;
};
```

CDIOCPLAYMSF (struct ioc_play_msf) Start audio playback given a ‘minutes-seconds-frames’ address and length. The structure is defined as follows:

```
struct ioc_play_msf
{
        u_char  start_m;
        u_char  start_s;
        u_char  start_f;
        u_char  end_m;
        u_char  end_s;
        u_char  end_f;
};
```

CDIOCREADSUBCHANNEL (struct ioc_read_subchannel) Read information from the subchannel at the location specified by this structure:

```
struct ioc_read_subchannel {
        u_char address_format;
#define CD_LBA_FORMAT 1
#define CD_MSF_FORMAT 2
        u_char data_format;
#define CD_SUBQ_DATA            0
#define CD_CURRENT_POSITION     1
#define CD_MEDIA_CATALOG        2
#define CD_TRACK_INFO           3
        u_char track;
        int     data_len;
        struct  cd_sub_channel_info *data;
};
```

CDIOREADTOCHEADER (struct ioc_toc_header) Return summary information about the table of

contents for the mounted CD-ROM. The information is returned into the following structure:

```
struct ioc_toc_header {
        u_short len;
        u_char  starting_track;
        u_char  ending_track;
};
```

CDIOREADTOCENTRYS (struct ioc_read_toc_entry) Return information from the table of contents entries mentioned. (Yes, this command name is misspelled.) The argument structure is defined as follows:

```
struct ioc_read_toc_entry {
        u_char     address_format;
        u_char     starting_track;
        u_short    data_len;
        struct  cd_toc_entry *data;
};
```

The requested data is written into an area of size data_len and pointed to by data.

CDIOCSETPATCH (struct ioc_patch) Attach various audio channels to various output channels. The argument structure is defined thusly:

```
struct ioc_patch {
        u_char     patch[4];
        /* one for each channel */
};
```

CDIOCGETVOL

CDIOCSETVOL (struct ioc_vol) Get (set) information about the volume settings of the output channels. The argument structure is as follows:

```
struct      ioc_vol
{
        u_char     vol[4];
        /* one for each channel */
};
```

CDIOCSETMONO	Patch all output channels to all source channels.
CDIOCSETSTEREO	Patch left source channel to the left output channel and the right source channel to the right output channel.
CDIOCSETMUTE	Mute output without changing the volume settings.
CDIOCSETLEFT	
CDIOCSETRIGHT	Attach both output channels to the left (right) source channel.
CDIOCSETDEBUG	
CDIOCCLRDEBUG	Turn on (off) debugging for the appropriate device.
CDIOCPAUSE	
CDIOCRESUME	Pause (resume) audio play, without resetting the location of the read-head.
CDIOCRESET	Reset the drive.
CDIOCSTART	
CDIOCSTOP	Tell the drive to spin-up (-down) the CD-ROM.
CDIOCALLOW	
CDIOCPREVENT	Tell the drive to allow (prevent) manual ejection of the CD-ROM disc. Not all drives support this feature.
CDIOCEJECT	Eject the CD-ROM.
CDIOCCLOSE	Tell the drive to close its door and load the media. Not all drives support this feature.

NOTES

When a CD-ROM is changed in a drive controlled by the **cd** driver, then the act of changing the media will invalidate the disklabel and information held within the kernel. To stop corruption, all accesses to the device will be discarded until there are no more open file descriptors referencing the device. During

this period, all new open attempts will be rejected. When no more open file descriptors reference the device, the first next open will load a new set of parameters (including disklabel) for the drive.

The audio code in the **cd** driver only support SCSI-2 standard audio commands. As many CD-ROM manufacturers have not followed the standard, there are many CD-ROM drives for which audio will not work. Some work is planned to support some of the more common ‘broken’ CD-ROM drives; however, this is not yet under way.

SYSCTL VARIABLES

The following variables are available as both sysctl(8) variables and loader(8) tunables:

kern.cam.cd.retry_count

This variable determines how many times the **cd** driver will retry a READ or WRITE command. This does not affect the number of retries used during probe time or for the **cd** driver dump routine. This value currently defaults to 4.

kern.cam.cd.%d.minimum_cmd_size

The **cd** driver attempts to automatically determine whether the drive it is talking to supports 6 byte or 10 byte MODE SENSE/MODE SELECT operations. Many SCSI drives only support 6 byte commands, and ATAPI drives only support 10 byte commands. The **cd** driver first attempts to determine whether the protocol in use typically supports 6 byte commands by issuing a CAM Path Inquiry CCB. It will then default to 6 byte or 10 byte commands as appropriate. After that, the **cd** driver defaults to using 6 byte commands (assuming the protocol the drive speaks claims to support 6 byte commands), until one fails with a SCSI ILLEGAL REQUEST error. Then it tries the 10 byte version of the command to see if that works instead. Users can change the default via per-drive sysctl variables and loader tunables. Where "%d" is the unit number of the drive in question. Valid minimum command sizes are 6 and 10. Any value above 6 will be rounded to 10, and any value below 6 will be rounded to 6.

FILES

/dev/cd[0-9][a-h] raw mode CD-ROM devices

DIAGNOSTICS

None.

SEE ALSO

cam(4), da(4), cd9660(5), disklabel(8), cd(9)

HISTORY

This **cd** driver is based upon the **cd** driver written by Julian Elischer, which appeared in 386BSD-0.1. The CAM version of the **cd** driver was written by Kenneth Merry and first appeared in FreeBSD 3.0.

BUGS

The names of the structures used for the third argument to **ioctl**() were poorly chosen, and a number of spelling errors have survived in the names of the **ioctl**() commands.