NAME

readcd - read or write data Compact Discs or related media

SYNOPSIS

readcd [**dev=***device*][*options*]

DESCRIPTION

readcd is used to read or write Compact Discs.

Device naming

Most users do not need to care about device naming at all. If no **dev=** option was specified, **readcd** implements **auto target** support and automagically finds the drive in case that exactly one CD-ROM type drive is available in the system. In case that more than one CD-ROM type drive exists on the system, a list of possible device name parameters may be retrieved with **readcd -scanbus** or from the target example from the output of **readcd dev=help**, then the **dev=** parameter may be set based on the device listing.

The *device* parameter to the **dev=** option explained below refers to the **SCSI CAM** standard notation for *scsibus*/*target*/*lun* of the CD/DVD/BluRay-Recorder. If a file /etc/default/cdrecord exists, the parameter to the **dev=** option may also be a drive name label in said file (see FILES section).

OPTIONS

If no options except the *dev=* option have been specified, **readcd** goes into interactive mode. Select a primary function and then follow the instructions.

Informative options

-help

display version information for **readcd** on standard output.

-version

Print version information and exit.

-v Increment the level of general verbosity by one. This is used e.g. to display the progress of the process.

Readcd functional options

-clone

Do a clone read. Read the CD with all sub-channel data and a full TOC. The full TOC data will be

put into a file with similar name as with the **f=** option but the suffix **.toc** added.

Note that reading in **clone** mode results in having no error correction at sub-channel level. Even in the main data channel, there is less error correction than with other read modes. This results in a slightly quality degradation. Avoid copying audio CDs in **clone** mode for this reason.

-c2scan

Scans the whole CD or the range specified by the **sectors=***range* for C2 errors. C2 errors are errors that are uncorrectable after the second stage of the 24/28 + 28/32 Reed Solomon correction system at audio level (2352 bytes sector size). If an audio CD has C2 errors, interpolation is needed to hide the errors. If a data CD has C2 errors, these errors are in most cases corrected by the ECC/EDC code that makes 2352 bytes out of 2048 data bytes. The ECC/EDC code should be able to correct about 100 C2 error bytes per sector.

If you find C2 errors you may want to reduce the speed using the **speed=** option as C2 errors may be a result of dynamic unbalance on the medium.

-cxscan

Scans the whole CD or the range specified by the **sectors=***range* for C1/C2/CU errors. In non-verbose mode, only a summary is printed. With **-v**, a line for each non error free second is printed. with **-vv**, a line for each second is printed. This scan method only works for a few drives.

-edc-corr

In this mode, **readcd** reads CD data sectors in uncorrected audio mode and then tries to correct the data using the ECC/EDC decoder library from Heiko Eissfeldt. As this library implements looping over two layers of error correction, **readcd** may be able to correct more data than the firmware of the CD-ROM drive.

This option is currently experimental and only applicable with CD media and currently only supports plain 2048 Byte CD-ROM sectors.

f=*file*

Specify the filename where the output should be written or the input should be taken from. Using '-' as filename will cause **readcd** to use **stdout** resp. **stdin**.

-factor

Output the speed values for **meshpoints=**# as factor based on *single speed* of the current medium. This only works if **readcd** is able to determine the current medium type.

-fulltoc

Retrieve a full TOC from the current disk and print it in hex.

meshpoints=#

Print read-speed at # locations. The purpose of this option is to create a list of read speed values suitable for e.g. **gnuplot**. The speed values are calculated assuming that 1000 bytes are one kilobyte as documented in the SCSI standard. The output data created for this purpose is written to *stdout*.

-nocorr

Switch the drive into a mode where it ignores read errors in data sectors that are a result of uncorrectable ECC/EDC errors before reading. If **readcd** completes, the error recovery mode of the drive is switched back to the remembered old mode.

-noerror

Do not abort if the high level error checking in **readcd** found an uncorrectable error in the data stream.

-notrunc

Do not truncate the output file when opening it.

-overhead

Meter the SCSI command overhead time. This is done by executing several commands 1000 times and printing the total time used. If you divide the displayed times by 1000, you get the average overhead time for a single command.

-pi8scan

Scans the whole DVD or the range specified by the **sectors=***range* for **pisum8** errors. In non-verbose mode, only a summary is printed. With **-v**, a line for each non error free block of 8 * 32 kB is printed. with **-vv**, a line for each block of 8 * 32 kB is printed. This scan method only works for a few drives.

-pifscan

Scans the whole DVD or the range specified by the **sectors=***range* for **pif** errors. In non-verbose mode, only a summary is printed. With **-v**, a line for each non error free block of 32 kB is printed. with **-vv**, a line for each block of 32 kB is printed. This scan method only works for a few drives.

-plot

This option modified the behavior for **-cxscan**, **-pi8scan** and **-pifscan**. The output is better suited for gnuplot.

retries=*#*

Set the retry count for high level retries in **readcd** to #. The default is to do 128 retries which may be too much if you like to read a CD with many unreadable sectors.

sectors=*range*

Specify a sector range that should be read. The range is specified by the starting sector number, a minus sign and the ending sector number. The end sector is not included in the list, so **sectors**=0-0 will not read anything and may be used to check for a CD in the drive.

speed=#

Set the speed factor of the read or write process to #. # is an integer, representing a multiple of the audio speed. This is about 150 KB/s for CD-ROM and about 172 KB/s for CD-Audio. If no *speed* option is present, **readcd** will use maximum speed. Only MMC compliant drives will benefit from this option. The speed of non MMC drives is not changed.

Using a lower speed may increase the readability of a CD or DVD.

-w Switch to write mode. Writing is only possible to DVD-RAM media. For other media, use **cdrecord** instead. Note that **cdrecord** also supports to write DVD-RAM media.

If this option is not present, **readcd** reads from the specified device.

SCSI options

dev=*target*

Set the SCSI target for the CD/DVD/BluRay-Recorder, see notes above. A typical target device specification is **dev**=*1,6,0* . If a filename must be provided together with the numerical target specification, the filename is implementation specific. The correct filename in this case can be found in the system specific manuals of the target operating system. On a *FreeBSD* system without *CAM* support, you need to use the control device (e.g. */dev/rcd0.ctl*). A correct device specification in this case may be **dev**=*/dev/rcd0.ctl:@* .

General SCSI addressing

The *target device* to the **dev=** option refers to the **SCSI CAM** standard notation for *scsibus*/*target*/*lun* of the CD/DVD/BluRay-Recorder. Communication on *SunOS* is done with the SCSI general driver **scg.** Other operating systems are using a library simulation of this driver. Possible syntax is: **dev=** *scsibus,target,lun* or **dev=** *target,lun*. In the latter case, the CD/DVD/BluRay-Recorder has to be connected to the default SCSI bus of the machine. *Scsibus*, *target* and *lun* are integer numbers. Some operating systems or SCSI transport implementations may require to specify a filename in addition. In this case the correct syntax for the device is: **dev=** *devicename*:*scsibus,target,lun* or **dev=** *devicename*:*target,lun*. If the name of the device node that has been specified on such a system refers

to exactly one SCSI device, a shorthand in the form **dev=** *devicename*:@ or **dev=** *devicename*:@,*lun* may be used instead of **dev=** *devicename*:*scsibus,target,lun*.

Remote SCSI addressing

To access remote SCSI devices, you need to prepend the SCSI device name by a remote device indicator. The remote device indicator is either **REMOTE:***user@host:* or **REMOTE:***host:* A valid remote SCSI device name may be: **REMOTE:***user@host:* to allow remote SCSI bus scanning or **REMOTE:***user@host:1,0,0* to access the SCSI device at *host* connected to SCSI bus # 1,target 0, lun 0. In order to allow remote access to a specific *host*, the **rscsi**(1) program needs to be present and configured on the *host*.

Alternate SCSI transports

Cdrecord is completely based on **SCSI** commands but this is no problem as all CD/DVD/BluRay writers ever made use **SCSI** commands for the communication. Even **ATAPI** drives are just **SCSI** drives that inherently use the *ATA packet interface* as **SCSI** command transport layer build into the IDE (ATA) transport. You may need to specify an alternate transport layer on the command line if your OS does not implement a fully integrated kernel driver subsystem that allows one to access any drive using **SCSI** commands via a single unique user interface.

To access SCSI devices via alternate transport layers, you need to prepend the SCSI device name by a transport layer indicator. The transport layer indicator may be something like **USCSI:** or **ATAPI:**. To get a list of supported transport layers for your platform, use **dev=** *HELP*:

Portability Background

To make **readcd** portable to all UNIX platforms, the syntax **dev=** *devicename*:*scsibus,target,lun* is preferred as it hides OS specific knowledge about device names from the user. A specific OS may not necessarily support a way to specify a real device file name nor a way to specify *scsibus,target,lun*.

Scsibus 0 is the default SCSI bus on the machine. Watch the boot messages for more information or look into **/var/adm/messages** for more information about the SCSI configuration of your machine. If you have problems to figure out what values for *scsibus,target,lun* should be used, try the **-scanbus** option of **readcd** described below.

Using logical names for devices

If no *dev* option is present, **readcd** will try to get the device from the **CDR_DEVICE** environment.

If a file /etc/default/cdrecord exists, and if the argument to the **dev=** option or the **CDR_DEVICE** environment does not contain the characters ',', '/', '@' or ':', it is interpreted as a device label name that was defined in the file /etc/default/cdrecord (see FILES section).

Autotarget Mode

If no **dev=** option and no **CDR_DEVICE** environment is present, or if it only contains a transport specifyer but no address notation, **readcd** tries to scan the SCSI address space for CD-ROM drives. If exactly one is found, this is used by default.

debug=*#*, **-d**

Set the misc debug value to # (with debug=#) or increment the misc debug level by one (with -d). If you specify *-dd*, this equals to **debug=***2*. This may help to find problems while opening a driver for libscg. as well as with sector sizes and sector types. Using **-debug** slows down the process and may be the reason for a buffer underrun.

kdebug=*#*, **kd=***#*

Tell the **scg**-driver to modify the kernel debug value while SCSI commands are running.

-scanbus

Scan all SCSI devices on all SCSI busses and print the inquiry strings. This option may be used to find SCSI address of the devices on a system. The numbers printed out as labels are computed by: **bus * 100 + target**

scgopts=*list*

A comma separated list of SCSI options that are handled by libscg. The implemented options may be updated independently from applications. Currently, one option: **ignore-resid** is supported to work around a Linux kernel bug.

-silent, **-s**

Do not print out a status report for failed SCSI commands.

spt=*#*

Set the maximum number of sectors per transfer.

timeout=*#*

Set the default SCSI command timeout value to # seconds. The default SCSI command timeout is the minimum timeout used for sending SCSI commands. If a SCSI command fails due to a timeout, you may try to raise the default SCSI command timeout above the timeout value of the failed command. If the command runs correctly with a raised command timeout, please report the better timeout value and the corresponding command to the author of the program. If no *timeout* option is present, a default timeout of 40 seconds is used.

ts=*#*

Set the maximum transfer size for a single SCSI command to #. The syntax for the **ts=** option is

the same as for cdrecord fs=# or sdd bs=#.

If no **ts=** option has been specified, **readcd** defaults to a transfer size of 256 kB. If libscg gets lower values from the operating system, the value is reduced to the maximum value that is possible with the current operating system. Sometimes, it may help to further reduce the transfer size or to enhance it, but note that it may take a long time to find a better value by experimenting with the **ts=** option.

-V Increment the verbose level with respect of SCSI command transport by one. This helps to debug problems during the process, that occur in the CD-Recorder. If you get incomprehensible error messages you should use this flag to get more detailed output. **-VV** will show data buffer content in addition. Using **-V** or **-VV** slows down the process.

EXAMPLES

For all examples below, it will be assumed that the drive is connected to the primary SCSI bus of the machine. The SCSI target id is set to 2.

To read the complete media from a CD-ROM writing the data to the file *cdimage.raw*:

```
readcd dev=2,0 f=cdimage.raw
```

To read sectors from range 150 ... 10000 from a CD-ROM writing the data to the file *cdimage.raw*:

```
readcd dev=2,0 sectors=150-10000 f=cdimage.raw
```

To write the data from the file *cdimage.raw* (e.g. a filesystem image from **mkisofs**) to a DVD-RAM, call:

```
readcd dev=2,0 -w f=cdimage.raw
```

ENVIRONMENT

RSH

If the **RSH** environment is present, the remote connection will not be created via **rcmd**(3) but by calling the program pointed to by **RSH**. Use e.g. **RSH=**/usr/bin/ssh to create a secure shell connection.

Note that this forces **cdrecord** to create a pipe to the **rsh(1)** program and disallows **cdrecord** to directly access the network socket to the remote server. This makes it impossible to set up performance parameters and slows down the connection compared to a **root** initiated **rcmd(3)**

connection.

RSCSI

If the **RSCSI** environment is present, the remote SCSI server will not be the program **/opt/schily/sbin/rscsi** but the program pointed to by **RSCSI**. Note that the remote SCSI server program name will be ignored if you log in using an account that has been created with a remote SCSI server program as login shell.

EXIT STATUS

The following exit codes are used:

0 No error appeared.

-1 A specific error appeared. This may be a usage error caused by an illegal command line or another error with a problem specific error message from **readcd**.

-2 An unspecified error appeared during the process of talking to the drive. See SCSI error message for more information. The section **DIAGNOSTICS** below contains an explanation on how to read SCSI error messages.

Note that older operating systems and older shells may not support the full 32 bit range of the exit code and mask the value with 0xFF. This results in shortened exit codes in the range **0**..**255** where **-1** is mapped to **255**.

FILES

SEE ALSO

cdrecord(1), **mkisofs**(8), **rcmd**(3), **ssh**(1).

NOTES

If you don't want to allow users to become root on your system, **readcd** may safely be installed suid root. This allows all users or a group of users with no root privileges to use **readcd. Readcd** in this case will only allow access to CD-ROM type drives- To give all user access to use **readcd,** enter:

```
chown root /usr/local/bin/readcd
chmod 4711 /usr/local/bin/readcd
```

To give a restricted group of users access to **readcd** enter:

```
chown root /usr/local/bin/readcd
chgrp cdburners /usr/local/bin/readcd
chmod 4710 /usr/local/bin/readcd
```

and add a group *cdburners* on your system.

Never give write permissions for non root users to the */dev/scg?* devices unless you would allow anybody to read/write/format all your disks.

You should not connect old drives that do not support disconnect/reconnect to either the SCSI bus that is connected to the CD-Recorder or the source disk.

When using **readcd** with the **Linux SCSI generic driver.** You should note that **readcd** uses a layer, that tries to emulate the functionality of the scg driver on top of the drives of the local operating system. Unfortunately, the sg driver on **Linux** has several flaws:

- It cannot see if a SCSI command could not be sent at all.
- It cannot get the SCSI status byte. **Readcd** for that reason cannot report failing SCSI commands in some situations.
- It cannot get real DMA count of transfer. **Readcd** cannot tell you if there is an DMA residual count.
- It cannot get number of bytes valid in auto sense data. **Readcd** cannot tell you if device transfers no sense data at all.
- It fetches to few data in auto request sense (CCS/SCSI-2/SCSI-3 needs >= 18).

DIAGNOSTICS

A typical error message for a SCSI command looks like:

```
readcd: I/O error. test unit ready: scsi sendcmd: no error
CDB:  00 20 00 00 00 00
status: 0x2 (CHECK CONDITION)
Sense Bytes: 70 00 05 00 00 00 00 0A 00 00 00 00 25 00 00 00 00 00
Sense Key: 0x5 Illegal Request, Segment 0
Sense Code: 0x25 Qual 0x00 (logical unit not supported) Fru 0x0
Sense flags: Blk 0 (not valid)
```

cmd finished after 0.002s timeout 40s

The first line gives information about the transport of the command. The text after the first colon gives the error text for the system call from the view of the kernel. It usually is: **I/O error** unless other problems happen. The next words contain a short description for the SCSI command that fails. The rest of the line tells you if there were any problems for the transport of the command over the SCSI bus. **fatal error** means that it was not possible to transport the command (i.e. no device present at the requested SCSI address).

The second line prints the SCSI command descriptor block for the failed command.

The third line gives information on the SCSI status code returned by the command, if the transport of the command succeeds. This is error information from the SCSI device.

The fourth line is a hex dump of the auto request sense information for the command.

The fifth line is the error text for the sense key if available, followed by the segment number that is only valid if the command was a *copy* command. If the error message is not directly related to the current command, the text *deferred error* is appended.

The sixth line is the error text for the sense code and the sense qualifier if available. If the type of the device is known, the sense data is decoded from tables in *scsierrs.c* . The text is followed by the error value for a field replaceable unit.

The seventh line prints the block number that is related to the failed command and text for several error flags. The block number may not be valid.

The eight line reports the timeout set up for this command and the time that the command really needed to complete.

BUGS

None currently known.

Mail bugs and suggestions to **schilytools@mlists.in-berlin.de** or open a ticket at **https://codeberg.org/schilytools/schilytools/issues**.

The mailing list archive may be found at:

https://mlists.in-berlin.de/mailman/listinfo/schilytools-mlists.in-berlin.de.

AUTHORS

Joerg Schilling and the schilytools project authors.

SOURCE DOWNLOAD

The source code for the **cdrtools** is included in the **schilytools** project and may be retrieved from the **schilytools** project at Codeberg at

https://codeberg.org/schilytools/schilytools.

The download directory is

https://codeberg.org/schilytools/schilytools/releases.

INTERFACE STABILITY

The interfaces provided by **readcd** are designed for long term stability. As **readcd** depends on interfaces provided by the underlying operating system, the stability of the interfaces offered by **readcd** depends on the interface stability of the OS interfaces. Modified interfaces in the OS may enforce modified interfaces in **readcd**.