GNU Video CD Info Program (*vcd-info*)

$Id: vcd-info.texi,v 1.10 2005/02/13 19:59:44 rocky Exp $
The GNU Video CD Authoring Tools.
for version 0.7.23, 13 February 2005

Rocky Bernstein et al.

Table of Contents

1 Introduction

Note in the guide, we use the term Video CD to mean some sort of Video Compact Disc. There are various formats of Video Compact Discs, the oldest format of which is called "VCD" and is followed by a version number: 1.0, 1.1 or 2.0. Hopefully it should also be clear by context whether we are referring to a class of Video Compact Discs or the specific older format called VCD *version-number*.

vcd-info's mission in life is to display the contents of a Video CD or CD image. Well, that's not actually true—if given an audio CD (CD-DA) or some other kind of CD, *vcd-info* will try to display some information about that, perhaps just a list of the tracks, with varying degrees of success; However Video CDs are what *vcd-info* understands best and it will you the most detail about. The disk "image" does not have to reside on a physical Compact Disc; *vcd-info* understand the BIN/CUE disk-image format used by a popular DOS/Window mastering tool.

You might use the information from *vcd-info* as a diagnostic or analytical tool. It may help you understand a particular Video CD that you have. Or it might be useful in conjunction with or in debugging a Video CD you are creating ("authoring"). But debugging is probably best left for another tool; *vcd-info* will only let you display information, you can't modify anything with this tool and it is not interactive.

Although I imagine most of the time this is how you will use *vcd-info*, there are some subsidiary uses.

vcd-info shows information about a Video CD and as such the information shown reflects, sort of, the way the information is stored. So in a sense *vcd-info* can teach you a little bit about how a Video CD is laid out.

In writing an Video CD input plugin for a media player, I wanted to get bits of information about the Video CD to display in the media player, such as the Video CD format used and the volume name. Before writing a C library for such purposes, a lower-tech, and more loosely-coupled way to do this was used. The program ran *vcd-info* with the specific information request and parsed the text output. For example, to get just the format that the Video CD on the"default" CD-ROM device is in you might run:

```
vcd-info --no-banner --show-format
```

To get the album information you might run:

```
vcd-info --no-banner --show-info album
```

After doing this for a while and seeing this was useful, I then wrote a Video CD "info" library. A library is much faster and furthermore much of the information is saved away in memory reducing the amount of disk I/O. Later it turned out that I could completely remove the CD-reading routines from the VCD module and get this from another library, but that's a different story. What's relevant is that after creating this library, I then modified *vcd-info* to use this library *libvcdinfo*. The *vcd-info* code then got shorter as some of it getting moved inside the library. And this program served as a test bed for the *libvcdinfo*.

So the last thing that *vcd-info* may be of useful for is if you want to see how to use the *libvcdinfo* library; you can consult the source program and see how *vcd-info* gets its information.

2 Command Options

There are a number of miscellaneous options, but there are two broad categories of options. There is a set of options for specifying what to dump, these all start *--show-* and another set of options which specify where the CD-ROM or CD-ROM disk-image is.

2.1 Options controlling where to read from

If you don't specify any place to read from *vcd-info* will try to read from the default device that may be appropriate for your OS.

There are two broad categories of input, reading from a device that contains a Compact Disc, and reading from a file on the filesystem that contains an image that could be burned into a Compact Disc. The option for the former category, reading a physical CD, is *--cdrom-device* (*-C*); options for the latter category, reading a file containing a CD disk image, are *--bin-file* (*-b*), *--cue-file* (*-c*), *--toc-file*, and *--nrg-file* (*-N*). There is one other option *--input* (*-i*) which blurs the distinction—it tries to figure out based on the name whether you are reading from a device containing a CD or not and do the appropriate thing. Of course, since it only makes sense to read from one place, these options are mutually exclusive; use only one option from this set.

2.1.1 Reading from a file containing a CD disk image

The option *--bin-file*, *--cue-file* or *--toc-file* (the short forms for the first two are these are *-b* and *-c* respectively) indicate to read from a file containing an image what might be burned onto Compact Disc. The format is expected to be in the BIN/CUE format is is accepted by many CD mastering tools.

If you use the *--bin-file* option, the file read is expected to be the ".BIN" part—this is the bulk of the data. If you use *--cue-file* the file read is a "cue sheet" a funny sort of CD mastering file. The corresponding BIN file (which again contains the bulk of the data) is not taken from the FILE statement inside the CUE file, but it is constructed from the CUE file name by replacing ".cue" by ".bin". The CUE file must have exactly one FILE statement.

If you do not specify a file name after *--bin-file*, a default name is used "videocd.bin" is probably used; likewise if you do not specify a filename after –cue-file a default name is used, probably "videocd.cue." These names I believe are the default output file names used when you use `vcdimager` or `vcdxbuild`.

If you use the *--toc-file* option, the file read is expected to be a CD-ROM disk image using for the format *cdrdao* uses.

If you use the *--nrg-file* option (short form *-N*), the file read is expected to be a CD-ROM disk image using a proprietary and unpublished format used by the Nero burning software. Because this is proprietary and not published, this code has been reverse engineered and we really can't do as good a job as we can with a cue-sheet file. It's better not to work with such things when possible.

2.1.2 Reading from CD-ROM Device

The option *--cdrom-device* or the short form *-C* is used when you want to read from a Compact Disc. The name given would be the device name. If you don't specify a device, a

default will be selected based on the Operating System or information that can be gleaned from the OS. For example on GNU/Linux, the default device may be */dev/cdrom*, while on Solaris it might be */vol/dev/aliases/cdrom0*.

2.1.3 Reading from wherever

The option *--input* or the short form of this *-i* tries to be intelligent by trying to figure out whether the thing you are referring to is a device or a bin/cue disk image of a CD.

Currently the algorithm it uses the one that the provided by from the underlying library: *libcdio*. It is pretty simple: if the thing referred to is what in Unix is called a "block" or "character" device (and there is such a notion in your OS), then it assumes a device name. When you do "long listing" ("ls -l") of the name and track down possible symbolic links you will see a "b" or "c" in the first column if the file is a block or character devices.

If the thing referred to in an *--input* option is a "regular" file then it is assumed you have a file image of a CD. If the filename ends (i.e. the extension is) ".bin" or ".BIN" then we assume a "BIN file" if the filename ends ".nrg or ".NRG we assume a Nero NRG disk image, and if the filename ends ".toc" or ".TOC" then we assume a *cdrdao* CD image.

Although this algorithm is subject to change and there may be more sophisticated or alternative methods, in practice I think this does pretty much what's expected.

2.2 Options controlling what do you want shown

vcd-info is pretty flexible about the information it can show you.

By default *vcd-info* will show just about everything it can about a Video CD image. There is additional information you can get using the option *--debug*, but this is of a debugging nature.

However if you specify something in particular to dump, for example to dump track information via command option *--show-tracks*, *vcd-info* switches behavior and shows you almost nothing except what you asked for. I wrote "almost nothing except" rather than "nothing except" because by default there still is a little bit of extra information that is shown: delimiters between requested sections and a banner. However there are options to turn these off too. You don't have to ask for just one section or one item of a section at a time, you can list many individual items or sections together.

Most of the individual items that can be shown are part of a larger group listed in Chapter 3 [Output Sections], page 10. For example to show the volume id of the Video CD image which resides in the PVD sections of output you would use the option *--show-pvd vol*.

To show everything in one of these large sections you give the section name followed by "-all." So to see the entire PVD section you would use the command option *--show-pvd-all*.

To see a list of the suboptions available use the sub-option "help"; for example to see the sub-options available for the PVD section, you would use the command option *--show-pvd help*.

2.3 A complete list of options

First some general-purpose options. These you might find on other programs.

`--help` This option give a list of options type *`--help`*. *`-?`* is an abbreviation however you will probably have to "escape" the question mark so that the shell doesn't interpret this; i.e. you may have to type *`-\?`*.

`--usage` This option shows an abbreviated usage message

`--debug` *`debug level`*
`-d` *`debug level`*
This option gives more verbose, debugging output. The higher the number given, the more the output. A value of 3 or greater includes library informative messages and a value of 4 or greater includes library debugging messages.

`--terse`
`-t` This option is the same as giving the combined options *`--no-header`* *`--no-banner`* *`--no-delimiter`*. This option can be useful if this program is getting called for information from another program and you don't want to have filter this information out.

`--quiet`
`-q` Show only critical messages

`--version`
`-V` Display version and copyright information (the stuff that is suppressed by *`--no-banner`* and exit after displaying this information. Sometimes, you just wanna know what version you got.

Next we have some miscellaneous options that don't fall into another particular category.

`--access-mode`
This option probably makes sense only if you are reading from a CD-ROM. This overrides the default way that we read from the CD-ROM. The possible values allowed depend on the CD-ROM driver. Some values which work on some drivers are "IOCTL", "READ_10," and "READ_CD."

`--no-banner`
`-B` Do not show program banner header and CVS Id line. This would be useful in a regression test program where you want to check the output but know that the version information is going to change over time.

`--no-delimiter`
`-D` Do not show delimiter lines around various the sections of Chapter 3 [Output Sections], page 10.

`--sector-2336`
This option only makes sense if you are reading from a BIN CD disk image. This indicates to *`vcd-info`* to assume a 2336-byte sector mode for image file. *Note: This option is slated to disappear.*

Use at most one option in an invocation from the following list. These options are described in general in Section 2.1 [Options controlling where to read from], page 2.

`--bin-file` [*filename*]
`-b` [*filename*]
: Specifies a BIN CD-ROM disk image to read from. See Section 2.1.1 [Reading from a file containing a CD disk image], page 2.

`--cdrom-device` [*device-name*]
`-C` [*device-name*]
: Specifies a CD-ROM device to read from. See Section 2.1.2 [Reading from CD-ROM Device], page 2.

`--cue-file` [*filename*]
`-c` [*filename*]
: Specifies a CUE file to read from. Preferred over a BIN file if available. See Section 2.1.1 [Reading from a file containing a CD disk image], page 2.

`--nrg-file` [*filename*]
`-N` [*filename*]
: Specifies a Nero NRG-format CD-ROM disk image to read from. See Section 2.1.1 [Reading from a file containing a CD disc image], page 2.

`--input` [*file or device name*]
`-i` [*file or device name*]
: This option is insensitive to whether the thing reading from is a CD-ROM device or a file. See Section 2.1.3 [Reading from wherever], page 3.

The options below control how much output you get and described in general in Chapter 3 [Output Sections], page 10.

`--no-ext-psd`
: If the Video CD is in format VCD 2.0, there can be an optional "extended" PSD which contains information in the PSD and area highlighting and selection information. To suppress reading and printing this information use this option. The option is only meaningful if the VCD format is 2.0.

`--show-entries-all`
`-E` : List everything in the Entries section. See Section 3.3 [Entries Section], page 12.

`--show-entries` *sub-option*
: Sub-options are: "count", "data", "id", "prof", or "vers".

  `count` : The number of entries. A number. See line 5 of the example in Section 3.3 [Entries Section], page 12.

  `data` : A complete list of the entries. See lines 6–9. of the example in Section 3.3 [Entries Section], page 12.

  `id` : An 8 characters name, probably "ENTRYVCD". The name may also be "ENTRYSVD" but this is deprecated. Some DVD players may require one or the other to play SVCDs. See line 2 of the example in Section 3.3 [Entries Section], page 12. The XML tag for this is <option name="svcd vcd30 entrysvd" value="true"/> or <option name="svcd vcd30 mpegav" value="true"/> or the tag can be left out in which case you'll get ENTRYVCD.

`prof` System profile tag. See line 4 of the example in Section 3.3 [Entries Section], page 12. If the format is HQSVCD the value will be 1. If the format is SVCD the value will be 0.

`vers` A 2-hex digit version number. See line 3 of the example in Section 3.3 [Entries Section], page 12.

`--show-filesystem`
`-F` List everything in the filesystem section. For information about what is in this section, see Section 3.2 [ISO-9660 section], page 10.

`--show-format`
`-f` Show the format that the Video CD is in. The value will be one of "VCD 1.1", "VCD 2.0", "SVCD", or "HQVCD". Note that *`vcd-info`* does not validate that the CD image conforms rigorously to any set of standards. However this does suggest what it might most closely conform to.

`--show-info-all`
List everything in the Info section. See Section 3.4 [Disc Information], page 12.

`--show-info` *`suboption`*
`-I` [*`sub-option`*]
Sub-options are: album, cc, count, id, ofm, lid2, lidn, pal, pbc, prof, psds, res, seg, segn, segs, spec, start, st2, vers, or vol.

`album` A description of the album; it could be an album id. The XML tag for this is <album-id>. It is a fixed-length blank-padded string of 16 characters. See line 5 of the example in Section 3.4 [Disc Information], page 12.

`cc` user data cc. See line 14 of the example in Section 3.4 [Disc Information], page 12.

`count` The number of disks in the volume, an integer. The XML tag for this is <volume-count>. See line 6 of the example in Section 3.4 [Disc Information], page 12.

`id` A string representing the Video CD format. "VIDEO_CD" for VCD 1.0, VCD 1.1, or VCD 2.0, "SUPERVCD" for SVCD, XVCD, CVD, and "HQ-VCD" for HQVCD. A Maximum of 8 characters. See line 2 of the example in Section 3.4 [Disc Information], page 12.

`ofm` offset multiplier. Value is given is in hexadecimal. See line 20 of the example in Section 3.4 [Disc Information], page 12.

`lid2` start lid #2. A boolean value. The XML tag for this is <next-volume-use-lid2>. See line 15 of the example in Section 3.4 [Info Section], page 12.

`lidn` Maximum LID number. See line 21 of the example in Section 3.4 [Info Section], page 12.

`pal` bit set of 98 PAL(=set)/NTSC flags. See lines 8–9 of the example in Section 3.4 [Disc Information], page 12.

`pbc` Show the field called "extended pbc" or "reserved2" A boolean value. If the image format is SVCD or HQVCD, this will be labeled "reserved2" rather than "extended pbc." Furthermore if the format is VCD 2.0, checks for for extended playback control files (EXT/LOT_X.VCD and EXT_PSD_X.VCD) will be made. If both files exist then VCD2.0 extended playback is probably used. See line 17 of Section 3.4 [Disc Information], page 12.

`prof` System profile tag. Used to define the set of mandatory parts to be applied for compatibility. Tag, 0x00 for "SUPERVCD", 0x01 for "HQ-VCD", 0x0*n* for VCD*x.n*. A hexadecimal number. See line 4 of the example in Section 3.4 [Disc Information], page 12.

`psds` PSD size. See line 18 of the example in Section 3.4 [Disc Information], page 12.

`res` Restriction number, e.g. "unsuitable for kids"; 0 is unrestricted, 1 is restricted category 1, 2 is restricted category 2, 3 is restricted category 3. The XML tag for this is <restriction>.

`seg` First segment MSF address. See line 19 of the example in Section 3.4 [Disc Information], page 12.

`segn` Max segment number. See line 22 of the example in Section 3.4 [Disc Information], page 12.

`segs` The name of file containing segment information. VCD/INFO.VCD or SVCD/INFO.SVD. See line 1 of the example in Section 3.4 [Disc Information], page 12.

`spec` Special info is encoded into the pictures. A boolean value. See line 13 of the example in Section 3.4 [Disc Information], page 12.

`start` start time offset. Only relevant if SVCD or HQVCD. The XML tag for this is <start-time-offset>.

`st2` start track #2. A boolean value. The XML tag for this is <next-volume-use-sequence2>. See line 16 of the example in Section 3.4 [Disc Information], page 12.

`vers` 0x01 for SVCD and VCD1.*x*; 0x02 – VCD2.0. A hexadecimal number. See line 3 of the example in Section 3.4 [Disc Information], page 12.

`vol` The volume number. The XML tag for this is <volume-number>. See line 7 of the example in Section 3.4 [Disc Information], page 12.

`--show-lot`
`-L`

List everything in the LOT section. See Section 3.5 [LID Offset Table (LOT) Section) Section], page 13.

`--show-psd`
`-p`

List everything in the PSD section. See Section 3.6 [Program Segment Descriptors (PSD) Section], page 13.

`--show-pvd-all`
: List everything in the PVD section. For an example of this section and some additional information, see the listing Section 3.1 [Primary Volume Descriptor (PVD) Section], page 10.

`--show-pvd` *suboption*
`-P` [*sub-option*]
: Sub-options are: app, id, iso, prep, pub, sys, vers, vol, volset, or xa.

  `app` The application id; a string of at most 128 characters and often indicates the program used to create the CD. The XML tag for this is <application-id>. See line 9 of the example in Section 3.1 [Primary Volume Descriptor (PVD) Section], page 10.

  `id` Some sort of string ID. See line 2 of the example in Section 3.1 [Primary Volume Descriptor (PVD) Section], page 10.

  `iso` ISO size in blocks. See line 10 of the example in Section 3.1 [Primary Volume Descriptor (PVD) Section], page 10.

  `prep` preparer id; a string of at most 128 characters. The XML tag for this is <preparer-id>. See line 7 of the example in Section 3.1 [Primary Volume Descriptor (PVD) Section], page 10.

  `pub` publisher id. The XML tag for this is <publisher-id>. See line 7 of the example in Section 3.1 [Primary Volume Descriptor (PVD) Section], page 10.

  `sys` The system ID; a string of at most 32 characters. The XML tag for this is <system-id>. See line 4 of the example in Section 3.1 [Primary Volume Descriptor (PVD) Section], page 10.

  `vers` The version ID. A decimal number. See line 3 of the example in Section 3.1 [Primary Volume Descriptor (PVD) Section], page 10.

  `vol` The volume ID. The XML tag for this is <volume-id>. See line 5 of the example in Section 3.1 [Primary Volume Descriptor (PVD) Section], page 10.

  `volset` Volume set id. See line 5 of the example in Section 3.1 [Primary Volume Descriptor (PVD) Section], page 10.

  `xa` A boolean indicating whether the XA marker is present or not. See line 11 of the example in Section 3.1 [Primary Volume Descriptor (PVD) Section], page 10.

`--show-scandata`
: This option is relevant only if the program streams are MPEG2's. That is the Video CD image is not VCD 1.0, VCD 1.1 VCD 2.0.

`-s`

`--show-search`
`-X` This option is relevant only if the program streams are MPEG2's. That is the VCD image is not VCD 1.0, VCD 1.1 VCD 2.0.

`--show-source`
`-S`

Just list out where you are reading from and the size of the image.

Here's an example of the output of this:

```
Source: image file '/src/dvdrip/dumbo/dumbo.cue'
Image size: 275792 sectors
```

`--show-tracks`
`-T`

List everything in the Tracks section. See Section 3.7 [Tracks Section], page 14.

3 Output Sections

When you run *vcd-info* to display the contents of an entire disk, unless you use the option *--no-delimiter* (*-D*) you'll notice large sections of output separated by a string of dashes like this:

```
--------------------------------------------------------------------------
```

These separate the major logical sections of a Video CD

Note that in example given in these sections, numbers at the beginning of a line are line numbers and are not part of the actual output. We use these to help refer to parts of the example.

3.1 Primary Volume Descriptor (PVD) Section

The first track of a Video CD is an ISO-9660 formatted track. As such it contains information defined by the ISO 9660 specification. The starting point for information from an ISO 9660 track is the PVD or Primary Volume Descriptor found at sector 16. A PVD is sort of like a "superblock" for a filesystem.

Here is part of sample output from the test PAL SVCD that can be found from: http://www.vcdimager.org/pub/vcdimager/examples/test_svcd/test_svcd_pal.zip

```
 1. ISO9660 primary volume descriptor
 2.  ID: 'CD001'
 3.  version: 1
 4.  system id: 'CD-RTOS CD-BRIDGE'
 5.  volume id: 'PAL_TEST_SVCD'
 6.  volumeset id: ''
 7.  publisher id: 'Herbert Valerio Riedel'
 8.  preparer id: 'GNU VCDIMAGER 0.7.10 LINUX-GNU/I386'
 9.  application id: ''
10. ISO size: 1426 blocks (logical blocksize: 2048 bytes)
11. XA marker present: yes
```

3.2 ISO-9660 section

This section contains something that looks pretty much like a directory listing obtained by interpreting the ISO-9660 filesystem information. Here's a sample:

```
/:
 d d---1xrxrxr 0 0 [fn 00] [LSN     18]      2048  .
 d d---1xrxrxr 0 0 [fn 00] [LSN     18]      2048  ..
 d d---1xrxrxr 0 0 [fn 00] [LSN     19]      2048  EXT
 d d---1xrxrxr 0 0 [fn 00] [LSN     20]      2048  MPEGAV
 d d---1xrxrxr 0 0 [fn 00] [LSN     21]      2048  VCD

/EXT/:
 d d---1xrxrxr 0 0 [fn 00] [LSN     19]      2048  .
 d d---1xrxrxr 0 0 [fn 00] [LSN     18]      2048  ..
 - ----1xrxrxr 0 0 [fn 01] [LSN    225]     65536  LOT_X.VCD;1
 - ----1xrxrxr 0 0 [fn 01] [LSN    257]      6800  PSD_X.VCD;1

/MPEGAV/:
 d d---1xrxrxr 0 0 [fn 00] [LSN     20]      2048  .
 d d---1xrxrxr 0 0 [fn 00] [LSN     18]      2048  ..
```

```
 - ---2-xrxrxr 0 0 [fn 01] [LSN    450] 639546208 (563593216)  AVSEQ01.DAT;1

/VCD/:
 d d---1xrxrxr 0 0 [fn 00] [LSN     21]      2048  .
 d d---1xrxrxr 0 0 [fn 00] [LSN     18]      2048  ..
 - ----1xrxrxr 0 0 [fn 00] [LSN    151]      2048  ENTRIES.VCD;1
 - ----1xrxrxr 0 0 [fn 00] [LSN    150]      2048  INFO.VCD;1
 - ----1xrxrxr 0 0 [fn 00] [LSN    152]     65536  LOT.VCD;1
 - ----1xrxrxr 0 0 [fn 00] [LSN    184]      2456  PSD.VCD;1
```

Note that all the filenames seem to end with *;1*. This is a "version" number of the file. ISO-9660 filename translation as might be found if the filesystem were mounted is not performed here; if it were, this would be stripped and probably the filenames would be in lowercase letters.

The "attribute" characters at the beginning of left-hand side of the listing, for the most part, looks like what you would get in a Unix listing. Except there are these 1's and 2's before what in Unix would be the user/group/other mode attributes. Those numbers indicate whether the track is "mode2 form1" or "mode2 form2." So the example above, AVSEQ01.DAT is in a mode2 form2 track while every other entry listed is in a mode2 form1 track. The movie part is is this mode2 form2 track while the remaining meta-data part is in a mode2 form1 track.

Another unusual thing about that single mode2 form2 track containing AVSEQ01.DAT, is that there seem to be two numbers listed before the AVSEQ01.DAT entry. All the other mode2 form1 entries have a single number listed beforehand. The first number (63954208) is the number of bytes that is occupied on the disk proper. The second number in parenthesis (563593216) is the number of bytes if you assumed the file had 2048-byte blocks rather than the 2334-byte mode2 format2 blocksize. If you were to mount the CD filesystem you might see the second smaller number in a listing there.

The exact meaning of the attributes section is listed below:

- The 1st character is either "d" if the entry is a directory, or "-" if not.
- The 2nd character is either "a" if the entry is CD-DA (audio), or "-" if not.
- The 3rd character is either "i" if the entry is interleaved, or "-" if not.
- The 4th character is either "2" if the entry is mode2 form2 or "-" if not.
- The 5th character is either "1" if the entry is mode2 form1 or "-" if not.Note that an entry will either be in mode2 form1 or mode form2. That is you will either see "2-" or "-1" in the 4th & 5th positions.
- The 6th and 7th characters refer to permissions for everyone while the
- the 8th and 9th characters refer to permissions for a group while, and
- the 10th and 11th characters refer to permissions for a user.

In each of these pairs the first character (6, 8, 10) is "x" if the entry is executable. For a directory this means the directory is allowed to be listed or "searched". The second character of a pair (7, 9, 11) is "r" if the entry is allowed to be read.

The item in brackets, e.g. *[LSN 450]* for the AVSEQ01.DAT entry is the "logical sector number" or the place on the CD where the entry starts.

3.3 Entries Section

Here is part of sample output from the Video CD used above.

```
 1. VCD/ENTRIES.VCD
 2. ID: 'ENTRYVCD'
 3. version: 0x02
 4. system profile tag: 0x00
 5. entries: 31
 6. ENTRY[00]: track# 1 (SEQUENCE[0]), LSN 450 (MSF 00:08:00)
 7. ENTRY[01]: track# 1 (SEQUENCE[0]), LSN 482 (MSF 00:08:32)
 8. ENTRY[02]: track# 1 (SEQUENCE[0]), LSN 5888 (MSF 01:20:38)
 9. ENTRY[03]: track# 1 (SEQUENCE[0]), LSN 8925 (MSF 02:01:00)
 ...
```

In the above example, in the first line corresponds the "filename" in the files section that starts at logical sector number 151. If this Video CD were in SVCD, XVCD, HQVCD or CVD format that line would probably say "SVCD/ENTRIES.SVD." As shown In line 5, I made 31 entries for the Video CD; only the first 4 are shown. Depending on how playback control is set up you should be able to randomly access up to 31 different playback locations on this Video CD. If you use a media player like xine and my plugin for it, there is a way jump to the beginning of each of these entries.

The "track #" and SEQUENCE number are about the same thing, the track being one more than the SEQUENCE number. Note on line 6 ENTRY[00] is also at logical sector number 450, the same as AVSEQ01.DAT. In other words "Entry 0" is the start of the movie.

The minutes, seconds, frames number (MSF 00:08:00) is equivalent to the LSN number. For those who wonder how to from the MSF to the LSN you basically multiply the seconds (8) by 75 since there are 75 frames in a second (note this has nothing to do with how fast this is displayed). If you get this you get 600 which is a bit large. But that's because the MSF number 00:00:00 is equal to 150 or the "pregap sector size."

3.4 Disc Information

Here is part of sample output from a SVCD. (This is not the same example as the Video CD used before.)

```
 1. SVCD/INFO.SVD
 2. ID: 'SUPERVCD'
 3. version: 0x01
 4. system profile tag: 0x00
 5. album id: '                '
 6. volume count: 3
 7. volume number: 2
 8. pal flags: 00000000 00000000 00000000 00000000 00000000 00000000
 9.  (bslbf)   00000000 00000000 00000000 00000000 00000000 00000000 00
10. flags:
11.  reserved1: no
12.  restriction: 0
13.  special info: no
14.  user data cc: no
15.  start lid #2: no
16.  start track #2: no
17.  reserved2: no
18. psd size: 2376
19. first segment addr: 00:05:00
20. offset multiplier: 0x08
```

```
21. maximum lid: 15
22. maximum segment number: 3
23. SEGMENT[1]: audio: no stream, video: NTSC still, continuation no, OGT substream: None
24. SEGMENT[2]: audio: no stream, video: NTSC still, continuation no, OGT substream: None
25. SEGMENT[3]: audio: no stream, video: NTSC still, continuation no, OGT substream: None
26. volume start time[0]: 0 secs
27. volume start time[1]: 0 secs
28. volume start time[2]: 0 secs
29. volume start time[3]: 0 secs
30. volume start time[4]: 0 secs
```

3.5 LID Offset Table (LOT) Section

Here is part of sample output from a SVCD of this section.

```
SVCD/LOT.SVD
 LID[1]: offset = 0 (0x0000)
 LID[2]: offset = 216 (0x001b)
 LID[3]: rejected
```

We have list id's (LIDs). The offset are offsets into the Program Segment descriptors described in Section 3.6 [PSD], page 13. LID 3 is rejected which means that you are not supposed to be able to jump to that entry directly but may reach it through program control.

3.6 Program Segment Descriptors (PSD) Section

Here is part of sample output from the test PAL SVCD that can be found from: http://www.vcdimager.org/pub/vcdimager/examples/test_svcd/test_svcd_pal.zip

```
SVCD/PSD.SVD
 PSD[00] (LID[1] @0x0000): play list descriptor
  NOI: 2 | LID#: 1 (rejected: no)
  prev: disabled | next: LID[2] @0x0003 | return: disabled
  playtime: 0/15s | wait: 0s | autowait: 0s
  play-item[0]: SEGMENT[8] (0x03ef)
  play-item[1]: SEGMENT[7] (0x03ee)
```

The first line SVCD/PSD.SVD again is the file entry that we are reading. Since the CD image is in the SVCD format have that particular name. If the CD image were in VCD 1.1 or VCD 2.0 format the line would read VCD/PSD.VCD.

Above NOI shows that there are two play-item entries, the individual entries are given below. We also see that this item is not "rejected" and thus could be accessed directly from a player.

The corresponding XML text for the above output is:

```
<playlist id="playlist-01">
  <next ref="playlist-02"/>
  <wait>0</wait>
  <autowait>0</autowait>
  <play-item ref="segment-0007"/>
  <play-item ref="segment-0006"/>
</playlist>
```

Some of the tag names like "play-item," "wait, and "autowait" given above match what are in the XML description. Although, you won't find "next," "prev" or "return" listed, the tag names are also the same. In the XML you are free to omit values and you will get default values put in; in *vcd-info* you see what has been filled in.

Below is another excerpt from that file.

```
PSD[03] (LID[4] @0x0007): selection list descriptor
Flags: 0x01 | NOS: 9 | BSN: 1 | LID: 4 (rejected: yes)
prev: disabled | next: disabled | return: disabled
default: LID[12] @0x0029 | timeout: LID[2] @0x0003
wait: 10 secs | loop: 1 (delayed: no)
play-item: SEGMENT[1] (0x03e8)
ofs[0]: LID[6] @0x0015
ofs[1]: LID[7] @0x0017
ofs[2]: LID[8] @0x0019
ofs[3]: LID[9] @0x001b
ofs[4]: LID[11] @0x0026
ofs[5]: LID[10] @0x001d
ofs[6]: LID[13] @0x002b
ofs[7]: disabled
ofs[8]: LID[14] @0x002e
prev_area: disabled | next_area: disabled
retn_area: disabled | default_area: disabled
area[0]: disabled
area[1]: disabled
area[2]: disabled
area[3]: disabled
area[4]: disabled
area[5]: disabled
area[6]: disabled
area[7]: disabled
area[8]: disabled
```

The XML excerpt that corresponds to this part is this:

```
<selection id="selection-01" rejected="true">
  <bsn>1</bsn>
  <default ref="playlist-10" x1="0" y1="0" x2="0" y2="0"/>
  <timeout ref="playlist-02"/>
  <wait>10</wait>
  <loop jump-timing="immediate">1</loop>
  <play-item ref="segment-0000"/>
  <select ref="playlist-05" x1="0" y1="0" x2="0" y2="0"/>
  <select ref="playlist-06" x1="0" y1="0" x2="0" y2="0"/>
  <select ref="playlist-07" x1="0" y1="0" x2="0" y2="0"/>
  <select ref="playlist-08" x1="0" y1="0" x2="0" y2="0"/>
  <select ref="playlist-09" x1="0" y1="0" x2="0" y2="0"/>
  <select ref="selection-02" x1="0" y1="0" x2="0" y2="0"/>
  <select ref="playlist-11" x1="0" y1="0" x2="0" y2="0"/>
  <select/>
  <select ref="playlist-12" x1="0" y1="0" x2="0" y2="0"/>
 </selection>
```

3.7 Tracks

Here is part of sample output from the test PAL SVCD that can be found from: http://www.vcdimager.org/pub/vcdimager/examples/test_svcd/test_svcd_pal.zip

```
1. SVCD/TRACKS.SVD
2.  ID: 'TRACKSVD'
3.  version: 0x01
4. tracks: 1
5.  track[00]: 00:59:61, audio: 2 streams, video: PAL stream, OGT stream:
    all available
6. CVD interpretation (probably)
```

```
 7. (track[00]: 00:59:61 (cumulated), audio: 00, ogt: de)
 8. ------------------------------------------------------------------------
 9. CD-ROM TRACKS
10. tracks:  1
11. track # 01, LSN   1576, MSF: 00:23:01, size:   13386240
12. leadout   : LSN   7486, MSF: 01:41:61, size:          0
```

Lines 1 and 2 you will see if the format is a SVCD, XVCD, or CVD. If the format is some sort of Video CD you will lines 1 and 2 will be:

```
VCD/INFO.VCD
 ID: 'VIDEO_CD'
```

There is one track on this SVCD with two audio streams (channel 0 is in German and stream 1 is in English but the language information can't be determined from information on the image). The video stream on the single stream is in PAL format and there is a subtitle. Actually the subtitle may be in one of at least two format, a format defined by Philips called OGT (Overlay Graphics Text) as it is listed above or it might be in another format used in CVD's.

On line 6, the CVD interpretation is a bit of a guess, and lines 6–7 only appears if the Video CD image is not VCD 1.*x* or VCD 2.0. There are all sorts of ways to violate standards and *vcd-info* doesn't do a rigorous job in determining if they have been violated. So it doesn't really know if the CD image is a CVD or not. Worse, we really don't have a firm grasp of what the CVD specifications are. Having written all this though, a guess is that if the subtitle's a private stream id in the MPEG file has hexadecimal number 0xde. This may be completely bogus though.

The bottom part of the display shows track info obtained in a lower-level way. The information above CDROM TRACKS comes from meta-data read in that IS09660 filesystem.

If you are reading from a CD-ROM device, this information comes from comes requests from the device to give track information. If what you are reading is a disk image, the only way to get this information has to come from a "cue sheet" which is why it is better to specify a CUE file than a BIN file.

4 Some simple command invocations.

Perhaps all you want to do is dump the information about a CD that is currently sitting in your CD-ROM drive. If the CD is in the "default" location for your OS, then this might work:

```
vcd-info
```

It means the same thing as:

```
vcd-info --cdrom-device
```

and using the "short" option:

```
vcd-info -C
```

I have a combined CD-ROM and DVD drive called */dev/dvd*. I have another CD-ROM drive which goes under the name */dev/cdrom*, but I generally prefer to use the DVD/CD-ROM drive for reading since it doesn't seem to run as hot. Since that drive is not the "default" (*/dev/cdrom*) on my GNU/Linux box, I have to give a location. So this will work:

```
vcd-info -C=/dev/dvd
```

Or even better, I'll let *vcd-info* figure out it's a CD-ROM device:

```
vcd-info -i /dev/dvd
```

The equals sign before */dev/dvd* is optional in either format.

Let's say you just used vcdimager or vcdxbuild it created output to its "default" location (probably *videocd.cue* and *videocd.bin*. To dump out this out:

```
vcd-info --cue-file
```

But if you specified the output to go to say *nausicaa.bin* and *nausicaa.cue* you would have to specify that location like this:

```
vcd-info -c nausicaa.cue
```

Or again not having to worry about what type of file you could use the "intelligent" input option again as we did before:

```
vcd-info -i nausicaa.cue
```

To list all of the sub-options to choose from in the Primary Volume Descriptor section (PVD) try this:

```
vcd-info --show-pvd help
```

Okay, now for for some more obscure stuff. Let's say I'm writing a front-end GUI to do all sorts of stuff and I just need to pick out the format (VCD 1.1, VCD 2.0, SVCD, CVD, XVCD, or HQVCD) of the Video CD in the default drive. Since I have to read in this output, I don't want any extra lines not relevant other than the version.

```
vcd-info --no-banner --show-format
```

Or even shorter:

```
vcd-info -t -f
```

I want to see the filesystem info and track info for the CD in my default drive

```
vcd-info --show-tracks --show-filesystem
```

I want to see the filesystem info and track info for the CD in my default drive

And finally I want to see just the "data" part — that is the list of the entries section, no summary information like a count or ID tag.

```
vcd-info --show-entries data
```

5 History and Thanks

This program started its life under the name "vcddebug" by Herbert Valerio Riedel. The overall structure is still largely that of this program. However *vcddebug* dumps all information about a Video CD always. I needed to pick out individual pieces of information from this for a media player plugin. Initially I was running *vcddebug* and parsing output. It became clear that it would be convenient to add options to reduce the amount of output. Initially there were single options added such as to show the VCD format. However with the bulk of information that's that is available, a sub-option system became very useful; that was added.

I was unhappy about the name "vcddebug" since there really was no debugging involved. Furthermore getting changes back into *vcddebug* was going a bit slowly. So *vcddump* (the original name of this program) was started and that allowed me more freedom to change the program in more drastic ways. In particular, it was becoming clear to me that what I really needed was a library to give information about a Video CD and much of this code was "inlined" inside *vcddebug*. So I pulled pieces of that out and a library, *libvcdinfo*, was created. As a result *vcd-info* became a test-bed for this library.

In developing my VCD plugin, I used *vcd-info* quite a bit. One of the annoying things about *vcddebug* was it's verbosity in specifying where to read from; I was constantly typing "–cdrom-device=/dev/cdrom" when I felt that the program should be able to figure this out. The distinction between a "device" and a file containing CD image while it may be fascinating to the program, was just not of interest to me as a user. So the ability to figure out what seems to be a reasonable guess about what's being read, short options for the classes of input and optional names were then added.

When my VCD input plugin got good enough to read CD-images from a file, I realized that in order to recover track information one needs to consult the CUE file. So the ability to read this kind of file was added to the library routines, and *vcd-info* was adjusted accordingly. Likewise dumping out track information when a image is not a VCD was added. Later, I realized that CD information (and control) was a beast of its own outside of *vcddump*, *libvcdinfo* or even *vcdimager*. So with Herbert Valerio Riedel's blessing, *libcdio* and later *libiso9660* were started by splitting off code from the *vcdimager* package.

In writing a utility programs to show off *libcdio*, I came across Gerd Knorr's *cdinfo* and with his permission I modified it calling *cd-info* (so both programs could be available simultaneously). I then realized that if *vcddump* were renamed *vcd-info* it might make a stronger uniform connection between these two similar programs.

And the rest is history.

General Index

Option Index

Sub-Option Index

The body of this manual is set in
cmr10 at 10.95pt,
with headings in **cmb10 at 10.95pt**
and examples in `cmtt10 at 10.95pt`.
cmti10 at 10.95pt,
cmb10 at 10.95pt, and
cmsl10 at 10.95pt
are used for emphasis.