

---

# Ripping Audio CDs to the Ogg Vorbis Format

Paul Hoadley, Logic Squad

Copyright © 2003 Paul Hoadley

\$Date: 2003/07/11 07:19:48 \$

This document describes how to partially automate the process of ‘ripping’ an audio compact disc (CD) to the **Ogg Vorbis** compressed audio format under **FreeBSD**. This is very much a work-in-progress: there are bound to be better ways to do it. The main goal, however, is to describe a non-GUI-dependent process: the instructions in this article do not rely on any desktop environment. The only pre-requisite, of course, is physical access to the machine's CD-ROM drive. There should be no other constraints on the system (for example, the X Windows System is not required).

## 1. Process Overview

The process of ripping an audio CD into any compressed audio format usually involves (at least) the following steps:

1. Sampling the CD audio data and dumping it into an uncompressed format on disk
2. Encoding the uncompressed format into a compressed format.

In this document, we will consider some refinements, including:

- Automatically consulting an online database to retrieve metadata relating to the CD (including album title, artist, and track names)
- Automatically encoding, renaming and adding the metadata to each ripped track.

There are certainly more potential refinements. Ideally, one should be able to insert a CD, type a command, and have the tracks added to a music collection with no further intervention.

## 2. Sampling audio CD data

The audio data on a CD can be extracted track-by-track to uncompressed .WAV sound files using the **cdda2wav** utility. This is part of the **cdrtools** port, and can be installed by executing the following as the `root` user:

```
# cd /usr/ports/sysutils/cdrtools
# make install
```

You obviously need to have read access to the CD-ROM drive on your machine. Depending on the permissions set for the device in `/dev`, this may require you to remain as the `root` user for the next step.

The **cdda2wav** has many options, and you should read the manual page thoroughly. For the purposes of this document, the following will suffice to rip an entire CD to individual tracks. Create a directory for each disc, as the created filenames will be the same as the last run by default. Change to that directory and run:

```
# cdda2wav --device /dev/acd0a --cddb 0 --bulk
```

In this example, the CD-ROM drive is at `/dev/acd0a`—this *may* be different in your system. The option `--cddb 0` tells **cdda2wav** to consult the default online database for CD metadata and prompt the user if there is a choice. `--bulk` ensures the whole disc is ripped track-by-track into separate files.

**cdda2wav** will print out some informational messages about the disc and the tracks it finds. It may pause while contacting the online database. If the database contains more than one entry for a given disc ID (presumably because more than one person has submitted a slightly different track list—often just capitalisation or genre differences), **cdda2wav** will provide you with a list of options, for example:

```
3 entries found:
00: rock 9b0a410c Violent Femmes / Violent Femmes
01: rock 9e0a410c Violent Femmes
02: newage a00a410c Violent Femmes / Violent Femmes
03: ignore
please choose one (0-3): 0
```

At this point, choose one of the options. There is often nothing separating the options, other than the ‘genre’ field, which is information we will not be using anyway. In this case, I would type **0**. **cdda2wav** should then print out some preliminary information, and a track list:

```
CDINDEX discid: f4QJf7DxjEZlH5XvdcfkAwIghPc-
CDDb discid: 0x9b0a420c CDDbP titles: resolved
CD-Text: not detected
```

```
CD-Extra: not detected
Album title: 'Violent Femmes' [from Violent Femmes]
Track 1: 'Blister In The Sun'
Track 2: 'Kiss Off'
Track 3: 'Please Do Not Go'
Track 4: 'Add It Up'
Track 5: 'Confessions'
Track 6: 'Prove My Love'
Track 7: 'Promise'
Track 8: 'To The Kill'
Track 9: 'Gone Daddy Gone'
Track 10: 'Good Feeling'
Track 11: 'Ugly'
Track 12: 'Gimme The Car'
```

The sampling process will take a few minutes, and **cdda2wav** will keep you informed of its progress.

### 3. Encoding to Ogg Vorbis

This part of the process involves the **oggenc** program from the **vorbis-tools** port. As the **root** user, install the **vorbis-tools** port:

```
# cd /usr/ports/audio/vorbis-tools
# make install
```

We will also be performing an XSLT transformation on the XML data retrieved from the online database, so you will need an XSLT processor. The **libxslt** port provides the very fast **xsltproc**:

```
# cd /usr/ports/textproc/libxslt
# make install
```

Having installed the tools, the next step is to transform the XML data in the file `audio.cdindex` into a standard shell script that will perform the encoding work for us. Save the following simple XSLT code as `makesh.xml` in the same directory as the ripped `.WAV` files:

```
<?xml version="1.0" encoding="utf-8"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
version="1.0">

<xsl:output method="text"/>
<xsl:strip-space elements="*/>

<xsl:param name="quality" select="'6'"/>

<xsl:template match="CDInfo">
<xsl:text>#!/bin/sh
# Generated by makesh.xml
# Album: </xsl:text>
<xsl:value-of select="Title"/><xsl:text>
</xsl:text>
<xsl:apply-templates/>
</xsl:template>
```

```
<xsl:template match="SingleArtistCD">
<xsl:for-each select="Track">
<xsl:text>oggenc</xsl:text>
<xsl:text> --quality=</xsl:text><xsl:value-of select="$quality"/>
<xsl:text> --output=&quot;</xsl:text>
<xsl:number value="@Num" format="01"/>
<xsl:text>_</xsl:text>
<xsl:value-of select="translate(Name, ' ', '_')"/>
<xsl:text>.ogg</xsl:text>
<xsl:text>&quot;</xsl:text>
<xsl:text> --artist &quot;</xsl:text>
<xsl:value-of select=" ../Artist"/>
<xsl:text>&quot;</xsl:text>
<xsl:text> --title &quot;</xsl:text>
<xsl:value-of select="."/>
<xsl:text>&quot;</xsl:text>
<xsl:text> --album &quot;</xsl:text>
<xsl:value-of select="//Title"/>
<xsl:text>&quot;</xsl:text>
<xsl:text> --tracknum </xsl:text>
<xsl:value-of select="@Num"/>
<xsl:text> audio_</xsl:text><xsl:number value="@Num" format="01"/>
<xsl:text>.wav</xsl:text>
<xsl:text>
</xsl:text>
</xsl:for-each>
</xsl:template>

<xsl:template match="*">
</xsl:template>
</xsl:stylesheet>
```

The stylesheet is fairly primitive, and really just serves to re-arrange the information we already have into a series of shell script lines. Only the `--quality` option is parameterised here, so it could be over-ridden at the command line. Track numbers are encoded into the filenames—I like this because it preserves album ordering in a directory listing. Spaces are converted to underscores in filenames.

Now that access to the CD-ROM device has concluded, you can work as any non-root user again. To generate the shell script, type:

```
> xsltproc -o script.sh makesh.xsl audio.cdindex
```

This command tells **xsltproc** to transform the XML in `audio.cdindex` into the shell script `script.sh` using the XSLT in `makesh.xsl`. Before proceeding, take a look at the generated script to ensure it is satisfactory:

```
> more script.sh
```

Occasionally, for example if the data returned by the online database is incomplete, the script can be generated with missing parameters. You should see a series of command lines beginning with the **oggenc** command, and including information such as the song and album titles, the artist and the track number:

---

## Ripping Audio CDs

---

```
oggenc --quality=6 --output="01_Blister_In_The_Sun.ogg"  
--artist "Violent Femmes" --title "Blister In The Sun"  
--album "Violent Femmes" --tracknum 1 audio_01.wav  
oggenc --quality=6 --output="02_Kiss_Off.ogg"  
--artist "Violent Femmes" --title "Kiss Off"  
--album "Violent Femmes" --tracknum 2 audio_02.wav
```

The lines are split here for readability, but will be continuous, long lines in the script itself.

If `script.sh` looks good, change it to be executable, and run it:

```
> chmod u+x script  
> script.sh
```

This will start the encoding process. It will take a few minutes per track, dependent on the speed of your CPU. When the encoding is finished, you should have correctly named `.ogg` files in the directory:

```
> ls -l *.ogg  
-rw-r--r-- 1 paulh paulh 3070070 Jul 4 14:20 01_Blister_In_The_Sun.ogg  
-rw-r--r-- 1 paulh paulh 3583998 Jul 4 14:22 02_Kiss_Off.ogg  
-rw-r--r-- 1 paulh paulh 5288692 Jul 4 14:25 03_Please_Do_Not_Go.ogg  
-rw-r--r-- 1 paulh paulh 5782202 Jul 4 14:29 04_Add_It_Up.ogg  
-rw-r--r-- 1 paulh paulh 6940948 Jul 4 14:33 05_Confessions.ogg  
-rw-r--r-- 1 paulh paulh 3182951 Jul 4 14:35 06_Prove_My_Love.ogg  
-rw-r--r-- 1 paulh paulh 3283854 Jul 4 14:38 07_Promise.ogg  
-rw-r--r-- 1 paulh paulh 4786760 Jul 4 14:41 08_To_The_Kill.ogg  
-rw-r--r-- 1 paulh paulh 4130547 Jul 4 14:43 09_Gone_Daddy_Gone.ogg  
-rw-r--r-- 1 paulh paulh 4728298 Jul 4 14:46 10_Good_Feeling.ogg  
-rw-r--r-- 1 paulh paulh 3220215 Jul 4 14:48 11_Ugly.ogg  
-rw-r--r-- 1 paulh paulh 6402192 Jul 4 14:52 12_Gimme_The_Car.ogg
```

They should also contain the correct metadata from the original album:

```
> vorbiscomment 01_Blister_In_The_Sun.ogg  
title=Blister In The Sun  
artist=Violent Femmes  
album=Violent Femmes  
tracknumber=1
```

You can now safely delete the `.wav` files:

```
> rm audio_*
```

You may wish to keep the files `audio.cddb`, `audio.cdindex` and the generated `script.sh` for future reference.

## 4. Enhancements

There is certainly a lot of scope for enhancing this process. At least the following spring to mind:

- Certainly the XSLT script should be kept somewhere more central. For example, I rip my CDs to `audio/cds` as a subdirectory of my home directory, so `make.sh.xsl` resides there on my system. Placing it in the working directory of the ripping process was a simplification for this article.
- The process seems *ideal* for further automation, perhaps via **make**. The sequence:

```
> cd ~/audio/cds; make rip encode clean
```

would be ideal. There should be nothing preventing the automatic creation of the whole directory hierarchy based on the metadata received about the album. *However*, on occasion the online database has no record for a particular disc ID. A **make**-based approach would need to be able to handle this, and may want to check that a sane script has been generated in the case of poor metadata being received. Just off the top of my head, I suspect that a double quote character in one of the title fields, for example, would break the simplistic approach described above.

## Contacting the author

The author of this document is **Paul Hoadley**. This document only describes what I did to rip some audio CDs to disk. Your mileage may vary. If you notice any errors in this document, or, better still, you have a system that is far superior to this, please **let me know**. I am interested in refining the general process described above, with the proviso that any refinements *don't* require use of the X Windows System or any X-related applications.