

iPod & iTunes HACKS™

100 Industrial-Strength Tips & Tools



O'REILLY®

Hadley Stern

HACK
#94

Manipulate Audio Using the Terminal

M V L

iTunes isn't the only way to encode your CDs. With Mac OS X, you can get to the heart of Unix to rip in alternative encoders such as LAME.

One of the greatest advantages of Apple's OS X operating system is its Unix core. Unix is a flexible environment (once you learn how to use it) that lets you get your hands dirty and solve problems when other applications fall short. While I love iTunes' powerful "jukebox" environment, I wish it allowed more options when it comes to audio encoding. The LAME encoder (<http://lame.sourceforge.net>; free) is the Internet standard for quality MP3 encoding. In the Usenet MP3 groups, you will find LAME in much wider use than any other codec, and your ears will hear the difference. Once I realized what I was missing by using the iTunes encoder, I set out to make using LAME easier.

Most command-line audio tools (such as the FLAC and LAME encoders) are designed to be used on only one file at a time, which makes batch processing tricky. A simple Unix shell script seemed the obvious solution to this problem. All big problems start small, however, and after I had written my batch LAME encoding script, I realized there were lots of little problems that I needed to solve in order to make CD archiving a more pleasant experience.

What began for me as a simple hack for batch encoding CDs and setting ID3 tags turned into a suite of programs aimed at streamlining the handling of MP3 and FLAC files from the command line. I chose the LAME (LAME Ain't an MP3 Encoder) and FLAC (Free Lossless Audio Codec: <http://flac.sourceforge.net/>; free) encoders, in part because they are both distributed under a version of the GNU General Public License (<http://www.gnu.org/licenses/licenses.html>), which keeps their development out in the open and ensures that end users are given rights to change the programs if they like. There are many advantages to both of these encoders, but audio quality and openness are chief among them.

Here, then, are the eight scripts to make your audio life easier:

lameit

Rips CDs to MP3 format using LAME

flacit

Rips CDs to FLAC format

id3hack

Uses the filename to set a track's name and number in its ID3 tag

vchack

Creates a Vorbis comment for a FLAC file using the *id3hack* script

Manipulate Audio Using the Terminal

vctool

Borrows the *id3tool* interface to set Vorbis comments in FLAC files

vcid3

Converts Vorbis comments in FLAC files to ID3 tags in MP3 files

flacmp3

Converts FLAC files to MP3 files

striptoc

Reformats a *cdrdao*-generated table of contents file for use with FLAC files

There are, of course, some binaries that you need in order to make these scripts work. I recommend you install them using *fink* (<http://fink.sourceforge.net>) or *darwinports* (<http://darwinports.opendarwin.org/>), or compile them by hand and put them in your *\$HOME/bin* directory:

- *cdrdao* (<http://cdrdao.sourceforge.net/>; for *striptoc* only)
- *flac* (<http://flac.sourceforge.net/>; for *flacit* and *flacmp3*):
- *id3tool* (<http://neko.hako.xware.cx/id3tool/>; for *id3hack* and *vcid3*)
- *lame* (<http://lame.sourceforge.net/>; for *lameit* and *flacmp3*)
- *metaflac* (<http://flac.sourceforge.net/>; for *flacmp3*, *vchack*, *vctool*, and *vcid3*)

To install each of these using *fink*, simply type:

```
% fink install <package-name>
```

Some packages might not be available under *fink*, or *fink* might use an outdated version, and you might need to compile the executable yourself. Instructions for doing this can be found on each program's web site and in the *README* file included with the source archive. Compilation usually involves running a *configure* script, followed by the *make* command. I recommend placing the compiled binaries in either */usr/local/bin* (for system-wide use) or *\$HOME/bin* (if you are not the system administrator or don't want to share the utilities with other users).

lameit

The first script uses LAME to rip a CD to the current directory. This process works best if you first set the CD metadata in iTunes. The easiest way to do this is to get the information from the Gracenote CDDB automatically by selecting Get CD Track Names from the Advanced menu in iTunes. If the CDDB information is incorrect, you can then edit it by hand within iTunes. Once you've done that (you might have to eject and reinsert the CD to make sure the info is updated), you're ready to start ripping.

The code. Type the following script and save it to a file called *lameit* in your *\$HOME/bin* directory:

```
#!/bin/sh
#
# lameit - rip a cd to lame-encoded mp3s
#
if [ "$1" ]
then
  for file in "$1"/[1-9]\ *.aiff
  do
    if [ -e "$file" ]
    then
      lame -h -m s -b 192 "$file" "0$(basename "$file" .aiff).mp3"
    else
      echo >&2 "No appropriate files exist in directory: "$1""
      exit 1
    fi
  done
  for file in "$1"/[1-9][0-9]\ *.aiff
  do
    if [ -e "$file" ]
    then
      lame -h -m s -b 192 "$file" "$(basename "$file" .aiff).mp3"
    fi
  done
else
  echo >&2 "Usage: "$(basename "$0")" /path/to/cd"
  exit 1
fi
```

The script simply checks for appropriate *.aiff* files (with the track number followed by a space in the filename) and encodes each one using LAME. In this case, the encoding is with a 192k constant bitrate, stereo.

Running the hack. Make the script executable by opening the Terminal application (*/Applications/Utilities/Terminal*) and typing the following on the command line:

```
% chmod +x lameit
```

You can then run the script this way:

```
% lameit /path/to/cd
```

Replace */path/to/cd* with the path to the CD you're interested in, which can be found in the */Volumes* directory.

You can modify the LAME command line in the script to suit your needs. Type `lame --help` in the Terminal for some guidelines on encoding options.

flacit

The next script does the same thing, but with FLAC, a lossless encoder, instead of LAME.

The code. Type the following script and save it to a file called *flacit* in your *\$HOME/bin* directory:

```
#!/bin/sh
#
# flacit - rip a cd to flac format
#
if [ "$1" ]
then
  for file in "$1"/[1-9]\ *.aiff
  do
    if [ -e "$file" ]
    then
      flac \
        --force-raw-format \
        --endian=little \
        --sign=signed \
        --channels=2 \
        --sample-rate=44100 \
        --bps=16 \
        --skip=20 \
        --output-name="0$(basename "$file" .aiff).flac" \
        "$file"
    else
      echo >&2 "No appropriate files exist in directory: "$1""
      exit 1
    fi
  done
  for file in "$1"/[1-9][0-9]\ *.aiff
  do
    if [ -e "$file" ]
    then
      flac \
        --force-raw-format \
        --endian=little \
        --sign=signed \
        --channels=2 \
        --sample-rate=44100 \
        --bps=16 \
        --skip=20 \
        --output-name="$(basename "$file" .aiff).flac" \
        "$file"
    fi
  done
else
  echo >&2 "Usage: "$(basename "$0")" /path/to/cd"
  exit 1
fi
```

Running the hack. Make the script executable by opening the Terminal application (*Applications/Utilities/Terminal*) and typing the following on the command line:

```
% chmod +x flacit
```

You can then run the script this way:

```
% flacit /path/to/cd
```

Replace */path/to/cd* with the path to the CD you're interested in, which can be found in the */Volumes* directory.

You can modify the FLAC command line in the script to suit your needs. Type `flac --help` in the Terminal for some guidelines on encoding options.

id3hack

Next comes the question of labeling the files. I use *id3tool* (<http://neko.hako.xware.cx/id3tool/>) to slap together ID3 tags before importing them into iTunes, because otherwise they get lost in my collection. *id3tool* works fine for labeling the artist, album, year, and genre, but setting the track number and song title can become tedious, so I whipped this little hack.



This hack works only if the files are named with the two-digit track number followed by its name—for example, *04 And Here We Test Our Powers of Observation.mp3*, *01 Moondance.mp3*, or *05 500 Miles.mp3*. You can specify as many files as you want on the command line. I usually just use the `*.mp3` wildcard.

The code. Type the following script and save it to a file called *id3hack* in your *\$HOME/bin* directory:

```
#!/bin/sh
#
# id3hack - add track names and numbers to id3 tags
#
if [ "$1" ]
then
  for file
  do
    if [ -e "$file" ]
    then
      id3tool \
        --set-title="$(echo "$file" | sed 's/...\.(\.*)\.mp3/\1/' ) \
          --set-track="$(echo "$file" | sed 's/\(\.\.\).*\/\1/' ) \
            "$file"
    else
      echo >&2 "No such file: "$1" -- skipping."
```

Manipulate Audio Using the Terminal

```

        fi
    done
else
    echo >&2 "Usage: "${basename "$0"}" INPUTFILE [...]"
    exit 1
fi

```

I used the Unix utility `sed` to extract the track name and number from the filename and set them as tags with `id3tool`. Of course, the script first checks to make sure that the files that you've given on the command line actually exist.

Running the hack. Make the script executable by opening the Terminal application (*/Applications/Utilities/Terminal*) and typing the following on the command line:

```
% chmod +x id3hack
```

You can then run the script by navigating to the directory containing the files you want to edit and typing:

```
% id3hack *.mp3
```

I used a wildcard here to apply to every MP3 file in the current directory, but I also could have supplied the filenames for each MP3 file. Either way, make sure the files exist in your current directory.

vchack

This is the same script as `id3hack`, except that it creates Vorbis comments for FLAC files instead of ID3 tags for MP3s.



This script uses *metaflac*, a tool for editing FLAC metadata that is included with FLAC.

The code. Type the following script and save it to a file called `vchack` in your `$HOME/bin` directory:

```

#!/bin/sh
#
# vchack - add track names and numbers to flac files
#
if [ "$1" ]
then
    for file
    do
        if [ -e "$file" ]
        then
            metaflac \
                --set-vc-field=TITLE="$(echo "$file" |

```

```

        sed 's/...\.(\.*)\.flac/\1/'" \
--set-vc-field=TRACKNUMBER="$(echo "$file" |
        sed 's/\(..\).*\/\1/' |
        sed 's/0\(\.\)/\1/'" \
"$file"
    else
        echo >&2 "No such file: "$1" -- skipping."
    fi
done
else
    echo >&2 "Usage: "${(basename "$0")}" INPUTFILE [...]"
    exit 1
fi

```

Again, this script is similar to *id3hack*. It uses *sed* and *metaflac* to pick apart the filename and assign pieces of it to metadata tags within the file.

Running the hack. Make the script executable by opening the Terminal (*Applications/Utilities/Terminal*) and typing the following on the command line:

```
% chmod +x vchack
```

You can then run the script from the directory containing the files you want to edit by typing the following:

```
% vchack *.flac
```

I used a wildcard here to apply to every FLAC file in the current directory, but I also could have supplied the filenames for each FLAC file. Either way, make sure the files exist in your current directory.

vctool

Vorbis comments can be tricky to work with. Out of frustration, I wrote a script that brought the *id3tool* interface over to the world of Vorbis comments and FLAC. Type *vctool -h* at the command line to get usage information.

The code. Type the following script and save it to a file called *vctool* in your *\$HOME/bin* directory:

```

#!/bin/sh
#
# vctool - set vorbis comments in flac files
#
if [ "$1" ]
then
    while getopts t:a:r:y:g:c:h option
    do
        case "$option" in
            t) TITLE="--set-vc-field=TITLE="$OPTARG"";;

```

Manipulate Audio Using the Terminal

```

a) ALBUM="--set-vc-field=ALBUM="$OPTARG"";;
r) ARTIST="--set-vc-field=ARTIST="$OPTARG"";;
y) DATE="--set-vc-field=DATE="$OPTARG"";;
g) GENRE="--set-vc-field=GENRE="$OPTARG"";;
c) TRACKNUMBER="--set-vc-field=TRACKNUMBER="$OPTARG"";;
h) echo "$$(basename "$0")" <options> <filename>"
    echo " -t WORD Sets the title to WORD"
    echo " -a WORD Sets the album to WORD"
    echo " -r WORD Sets the artist to WORD"
    echo " -y WORD Sets the date to WORD"
    echo " -g WORD Sets the genre to WORD"
    echo " -c WORD Sets the track number to WORD";;
    esac
done
shift $((OPTIND - 1))
for file
do
    if [ -e "$file" ]
    then
        for var in "$TITLE" "$ALBUM" "$ARTIST" "$DATE" "$GENRE" "$TRACKNUMBER"
        do
            if [ "$var" ]
            then
                metaflac "$var" "$file"
            fi
        done
    else
        echo >&2 "No such file: "$file" -- skipping."
    fi
done
else
    echo >&2 "Type "$$(basename "$0")" -h for help."
    exit 1
fi

```

In this script, each argument that you supply invokes a new instance of the program *metaflac*. I tried to make the script pass the arguments together to each file in one command but couldn't get it to work without *metaflac* assigning blank metadata tags. While the method used here is not ideal, it work's just fine; consider it a lazy hack.

Running the hack. Make the script executable by opening the Terminal application (*/Applications/Utilities/Terminal*) and typing the following on the command line:

```
% chmod +x vctool
```

Assigning metadata becomes much easier with *vctool*. Here is an example:

```

% vctool
Type vctool -h for help.
% vctool -h
vctool <options> <filename>

```

```

-t WORD    Sets the title to WORD
-a WORD    Sets the album to WORD
-r WORD    Sets the artist to WORD
-y WORD    Sets the date to WORD
-g WORD    Sets the genre to WORD
-c WORD    Sets the track number to WORD
% vctool -r "Archie Shepp" -a "Attica Blues" -y 1972 -g Jazz *.flac

```

Now, we've assigned artist, album, year, and genre metadata to every FLAC file in the current directory. That's a lot easier than typing this:

```

% metaflac --set-vc-field=ARTIST="Archie Shepp" *.flac
% metaflac --set-vc-field=ALBUM="Attica Blues" *.flac
% metaflac --set-vc-field=DATE=1972 *.flac
% metaflac --set-vc-field=GENRE=Jazz *.flac

```

vcid3

The *vcid3* script converts Vorbis comments to ID3 tags.

The code. Type the following script and save it to a file called *vcid3* in your *\$HOME/bin* directory:

```

#!/bin/sh
#
# vcid3 - convert vorbis comments to id3 tags
#
if [ -e "$1" ]
then
  if [ -e "$2" ]
  then
    TITLE="$(metaflac --show-vc-field=TITLE "$1" |
      sed 's/TITLE=\.*/\1/')"
    ARTIST="$(metaflac --show-vc-field=ARTIST "$1" |
      sed 's/ARTIST=\.*/\1/')"
    ALBUM="$(metaflac --show-vc-field=ALBUM "$1" |
      sed 's/ALBUM=\.*/\1/')"
    TRACK="$(metaflac --show-vc-field=TRACKNUMBER "$1" |
      sed 's/TRACKNUMBER=\.*/\1/')"
    YEAR="$(metaflac --show-vc-field=DATE "$1" |
      sed 's/DATE=\.*/\1/')"
    GENRE="$(metaflac --show-vc-field=GENRE "$1" |
      sed 's/GENRE=\.*/\1/')"
    if [ "$GENRE" ]
    then
      id3tool --set-genre-word="$GENRE" "$2"
    fi
    id3tool \
      --set-title="$TITLE" \
      --set-artist="$ARTIST" \
      --set-album="$ALBUM" \
      --set-track="$TRACK" \

```

Manipulate Audio Using the Terminal

```

        --set-year="$YEAR" \
        "$2"
    else
        echo >&2 "No such file: "$2""
        echo >&2 "Usage: "$("${basename "$0}")" FLACFILE MP3FILE"
        exit 1
    fi
else
    echo >&2 "No such file: "$1""
    echo >&2 "Usage: "$("${basename "$0}")" FLACFILE MP3FILE"
    exit 1
fi

```

This script avoids the resource fork issues of *vctool*, because *id3tool* does not assign blank tags. Rather, it simply leaves off tags that contain the empty string.

Running the hack. Make the script executable by opening the Terminal application (*/Applications/Utilities/Terminal*) and typing the following on the command line:

```
% chmod +x vcid3
```

To use the script, navigate to the directory containing the files you want to edit and simply supply the name of the FLAC file (containing the relevant metadata) and the name of the MP3 file (which will have the metadata assigned to it) on the command line:

```
% vcid3 "02 If You Want Me To Stay.flac" "02 If You Want Me To Stay.mp3"
```

flacmp3

Here's the way to get from FLAC to MP3 in one easy step. The script outputs the MP3 files to your current directory, but the FLAC files needn't be in your current directory.



This script performs the metadata conversion (without *id3tool*) side by side with the format conversion, so you won't need to do that separately.

The code. Type the following script and save it to a file called *flacmp3* in your *\$HOME/bin* directory:

```

#!/bin/sh
#
# flacmp3 - convert a flac file and its tag data to mp3/id3 format
#
if [ "$1" ]
then
    for file

```

```

do
  if [ -e "$file" ]
  then
    flac -c -d "$file" |
    lame -h -m s -b 192 \
      --tt "$(metaflac --show-vc-field=TITLE "$file" |
        sed 's/^TITLE=\.*/\1/'" \
      --ta "$(metaflac --show-vc-field=ARTIST "$file" |
        sed 's/^ARTIST=\.*/\1/'" \
      --tl "$(metaflac --show-vc-field=ALBUM "$file" |
        sed 's/^ALBUM=\.*/\1/'" \
      --ty "$(metaflac --show-vc-field=DATE "$file" |
        sed 's/^DATE=\.*/\1/'" \
      --tn "$(metaflac --show-vc-field=TRACKNUMBER "$file" |
        sed 's/^TRACKNUMBER=\.*/\1/'" \
      --tg "$(metaflac --show-vc-field=GENRE "$file" |
        sed 's/^GENRE=\.*/\1/'" \
      - "$(basename "$file" .flac).mp3"
  else
    echo >&2 "No such file: "$file" -- skipping."
  fi
done
else
  echo >&2 "Usage: "$(basename "$0")" FLACFILE [...]"
  exit 1
fi

```

I love this script. In one step, it performs a complete format conversion, including metadata, and it can do so on any number of files that you specify, even wildcards. Once again, sed to the rescue!

Running the hack. Make the script executable by opening the Terminal application (*/Applications/Utilities/Terminal*) and typing the following on the command line:

```
% chmod +x flacmp3
```

This script takes in FLAC files and spits out MP3 files. If you've assigned metadata to a FLAC file (using *vctool*, for example), it carries that information over to the MP3:

```
% flacmp3 *.flac
```

That's all it takes to do the job.

striptoc

Lastly, here's a script that takes a *cdrdao* table of contents (TOC) file and strips away all the unnecessary information in the file. *cdrdao* (<http://cdrdao.sourceforge.net>) is used for reading and writing raw CD data from the command line. Its most useful feature is its plain-text TOC files, which can be used to extract pregap information from source CDs. But the TOC files pre-

Manipulate Audio Using the Terminal

sume a single, huge data file, which is a really inconvenient way to archive a CD. This `awk` script takes a listing of FLAC files from the current directory and substitutes them for the track data file.



You'll have to decompress your FLACs before burning, of course.

The code. Here's an example of the *striptoc* script in action, using a CD that contains only two tracks:

```
% ls
01 So Long Eric.flac 02 Praying With Eric.flac Town Hall Concert.toc
% cat "Town Hall Concert.toc"
CD_DA

// Track 1
TRACK AUDIO
NO COPY
NO PRE_EMPHASIS
TWO_CHANNEL_AUDIO
FILE "data.wav" 0 17:48:03

// Track 2
TRACK AUDIO
NO COPY
NO PRE_EMPHASIS
TWO_CHANNEL_AUDIO
FILE "data.wav" 17:48:03 27:31:27
START 00:00:49

% striptoc "Town Hall Concert.toc" > "Town Hall Concert.toc.new"
% cat "Town Hall Concert.toc.new"
CD_DA

TRACK AUDIO
FILE "01 So Long Eric.wav" 0

TRACK AUDIO
PREGAP 00:00:49
FILE "02 Praying With Eric.wav" 0

% mv "Town Hall Concert.toc.new" "Town Hall Concert.toc"
%
```

As you can see, the file generated by *cdrdao* also explicitly states several defaults for each file. This script throws that information out, as well as any ISRC codes and catalog information (which this CD doesn't have). Here's the script that does all the work:

```
#!/usr/bin/awk -f
#
# striptoc - Reformat cdrdao toc files for use with individual track files.
#
BEGIN { print "CD_DA\n" }
{ FS = "\n"; RS = ""
  if ($2 == "TRACK AUDIO") {
    print $2
    if ($NF ~ /^START/) {
      sub(/^START/, "PREGAP", $NF)
      print $NF
    }
    FS = " "; RS = "\n"
    "ls *.flac" | getline file
    sub(/flac$/, "wav", file)
    print "FILE \"" file "\" o\n"
  }
}
```

awk can be harder to follow than a shell script, but basically, this script creates a new TOC file based on the pregap information given in the original file and the listing of *.flac* files in the current directory. It doesn't do any checks on the data beforehand, though, so make sure everything is in order before you run the script.

Running the hack. This AWK script sends its output to standard output, so you need to tell it where to put the newly generated file, and then (optionally) move that file back on top of the old file:

```
% striptoc Karma.toc > Karma.toc.new
mv Karma.toc.new Karma.toc
```

Final Thoughts

I hope you find these scripts useful. I think they really demonstrate the power and flexibility of Unix's programmer-friendly environment. Hopefully, they will inspire you to write scripts of your own to solve your little everyday problems.

—Chris Roose