Rip Audio CDs to MP3s

Converting audio CDs to MP3 consists of two steps:

1. rip the audio tracks from the cd and save them as wave file
2. convert the wave files to mp3 files

The tools required for these two steps are cdparanoia and lame. The graphical front-end ripperx can be used which also provides the option to add artist and title details and write mp3 tags.

Installation of the required tools (if graphical front-end is not required, remove ripperx):

```
apt-get install cdparanoia lame ripperx
```

To rip and convert a cd on the command line:

```
cdparanoia -B
#to extract only part of a track like time 0:13.13-1:13.00 from track 1, you
can use
#cdparanoia "1[:13.13]-1[1:13]"
for t in *.wav; do lame -b 192 -h -V 6 $t; done
```

Flac to MP3 conversion

To convert lossless FLAC files to mp3, the easiest option is using ffmpeg which will most likely already be installed. This script/command converts all flac files in a directory to MP3 V0 with variable bitrate between 220-260kbps. The resulting compacted audio file cannot be distinguished from the lossless version.

```
#!/bin/bash
for a in ./*.flac; do
  < /dev/null ffmpeg -i "$a" -qscale:a 0 "${a[@]/%flac/mp3}"
done
```

Trim silence from start/end

This reencodes the file

```
ffmpeg -i input.mp3 -af
silenceremove=1:0:-50dB:stop_periods=1:stop_duration=0:stop_threshold=-50dB
output.mp3
```

This copies the frames without quality loss:

trim_mp3_silence_copy.sh

```
#!/usr/bin/env bash
set -euo pipefail

# trim_mp3_silence_copy.sh
#
# Usage:
#   ./trim_mp3_silence_copy.sh input.mp3 [output.mp3]
#
# Env vars you can override:
#   NOISE_DB   (default: -40dB)  threshold for silencedetect
#   MIN_DUR    (default: 0.5)    minimum silence duration to consider
(seconds)

in="${1:-}"
out="${2:-}"

if [[ -z "${in}" ]]; then
  echo "Usage: $0 input.mp3 [output.mp3]" >&2
  exit 1
fi

if [[ ! -f "${in}" ]]; then
  echo "Input file not found: ${in}" >&2
  exit 1
fi

if [[ -z "${out}" ]]; then
  base="${in%.*}"
  out="${base}.trimmed.mp3"
fi

NOISE_DB="${NOISE_DB:--40dB}"
MIN_DUR="${MIN_DUR:-0.5}"

# Get full duration (seconds, as float)
duration="$(ffprobe -v error -show_entries format=duration -of
default=nw=1:nk=1 "$in")"

# Run silencedetect and capture lines
sd="$(
  ffmpeg -hide_banner -nostats -i "$in" \
    -af "silencedetect=noise=${NOISE_DB}:d=${MIN_DUR}" \
    -f null - 2>&1 | grep -E 'silence_(start|end):' || true
)"

if [[ -z "$sd" ]]; then
  echo "No silence detected (noise=${NOISE_DB}, d=${MIN_DUR}). Copying
original -> ${out}" >&2
  cp -f -- "$in" "$out"
```

```
  exit 0
fi

# First silence_end => start of audio
start_keep="$(awk '/silence_end:/ {print $NF; exit}' <<<"$sd")"

# Last silence_start => end of audio (trailing silence start)
end_keep="$(awk '/silence_start:/ {v=$NF} END{if(v!="") print v}'
<<<"$sd")"

# Fallbacks / sanity
if [[ -z "${start_keep:-}" ]]; then
  # No leading silence_end found; keep from 0
  start_keep="0"
fi
if [[ -z "${end_keep:-}" ]]; then
  # No trailing silence_start found; keep to full duration
  end_keep="$duration"
fi

# Ensure start < end (handle edge cases)
ok="$(
  awk -v s="$start_keep" -v e="$end_keep" 'BEGIN{ if (s+0 < e+0) print
"1"; else print "0"; }'
)"

if [[ "$ok" != "1" ]]; then
  echo "Could not determine a valid trim window." >&2
  echo "Detected start_keep=${start_keep}, end_keep=${end_keep},
duration=${duration}" >&2
  echo "Silencedetect output:" >&2
  echo "$sd" >&2
  exit 2
fi

echo "Input:  $in"
echo "Output: $out"
echo "Keep:   ${start_keep}s -> ${end_keep}s (duration ~ $(awk -v
s="$start_keep" -v e="$end_keep" 'BEGIN{printf "%.3f", (e-s)}'))"
echo "Detect: noise=${NOISE_DB}, min_silence=${MIN_DUR}s"

# Stream-copy trim (no re-encode). Use -ss/-to after -i for more
accurate seeking while still copying.
ffmpeg -hide_banner -y -i "$in" -ss "$start_keep" -to "$end_keep" -c
copy "$out"
```

Then use as

```
chmod +x trim_mp3_silence_copy.sh
./trim_mp3_silence_copy.sh input.mp3
```

```
# or
NOISE_DB=-35dB MIN_DUR=0.3 ./trim_mp3_silence_copy.sh input.mp3 out.mp3
```