MP3Gain

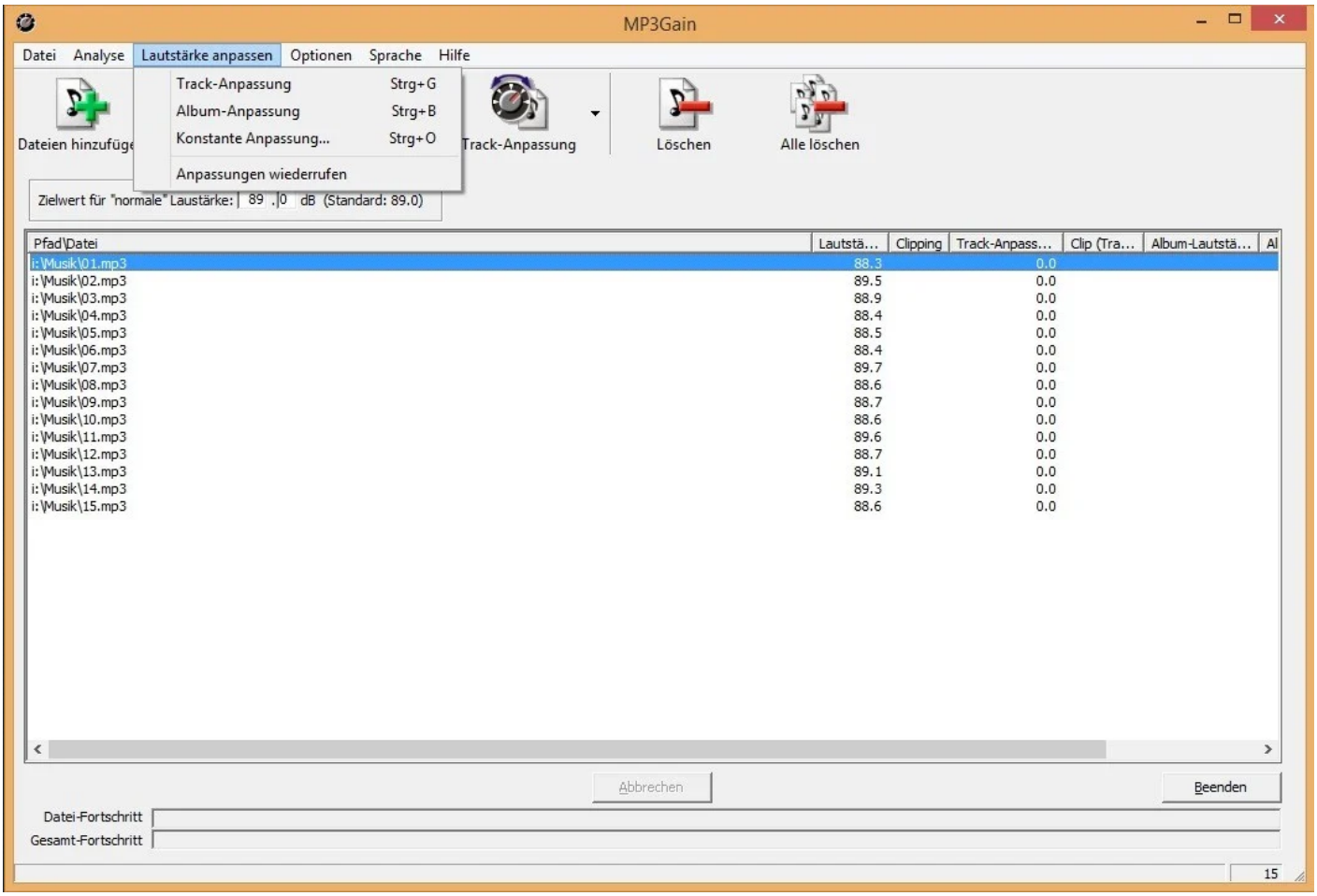
Audiodateien verlustfrei normalisieren. Die Lautstärke auf einen festen Wert oder Wert des Albums anpassen. Einzelne Dateien oder ganze Verzeichnisse. Bereits normalisierte Dateien werden erkannt und übersprungen.

Installation

MP3Gain steht für mehrere Plattformen zum Download bereit. Im folgenden wird kurz gezeigt, woher die Windows und Linux Varianten bezogen werden können.

Windows

Für Windows gibt es MP3Gain als GUI Anwendung z. B. auf Heise MP3Gain Windows Installer. Die Installation ist simpel, weiter, weiter, fertigstellen. Anschließend die Anwendung einfach gestartet werden, ein Verzeichnis oder einzelne Dateien auswählen und die Lautstärke anpassen lassen.



Linux (GUI)

Für Linux gibt es auch eine GUI Installation, die jedoch noch nicht getestet wurde. Näheres hierzu kann auf MP3Gain-GUI Ubuntuusers nachgelesen werden.

Linux (Terminal)

Der Vorteil bei der Installation der Terminal Anwendung ist, dass sich die Schritte zum Anpassen der Lautstärke hervorragend automatisieren lassen, z. B. als Cronjob oder nachgelagert nach Kopiervorgängen. Die Installation ist bei den meisten Linux Distributionen mit einem Paketmanager schnell erledigt, weil in der Bibliothek verfügbar.

Für APT ist das Kommando wie folgt.

```
sudo apt update
sudo apt -y install mp3gain
```

Anschließend kann das Programm über die Bash mit `mp3gain` aufgerufen werden.

Weitere Infos zu der Anwendung finden sich hier: MP3Gain Ubuntuusers

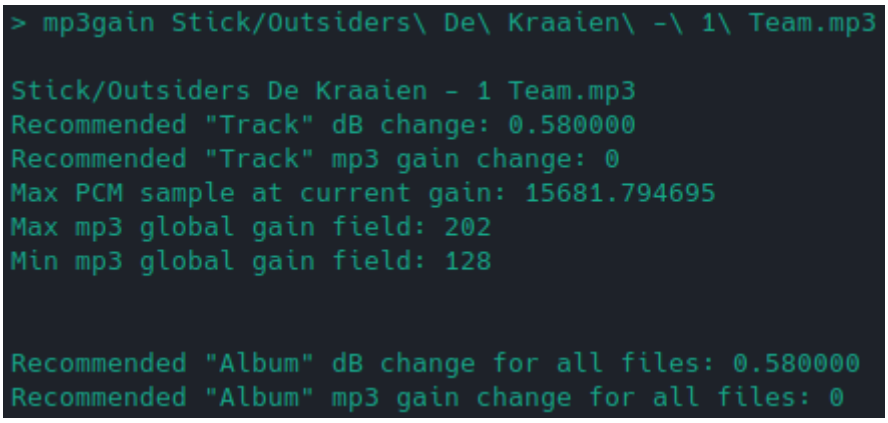
Normalisieren per Bash

Der Vorteil der Installation der Terminal-Variante von mp3gain (unter Linux der Standard) ist, dass sich Aufgaben wie z. B. das Normalisieren von Ordnern wunderbar automatisieren lassen.

Einzelne Dateien

Eine einzelne Datei lässt sich mit folgendem Befehl an die Standardlautstärke anpassen.

```
mp3gain -r /Pfad/zur/Musik/Songname.mp3
```



Ordner

Statt alle Dateien einzeln anzugeben, können auch alle Dateien und Unterordner in einem Ordner automatisch analysiert und normalisiert werden. Dies geht mit folgendem Befehl. Hier wurde zusätzlich noch der Filter auf `.mp3` Dateien hinzugefügt, dieser kann auch entfernt werden, um alle Dateien zu analysieren und zu normalisieren, sofern möglich.

```
mp3gain -r /Pfad/zur/Musik/*.mp3
```

Order mit Unterordner

Wenn nicht nur die Dateien im angegebenen Ordner normalisiert werden sollen, sondern auch die Dateien in den Unterordnern dieses Ordners, so ist dies mit dem folgenden Befehl möglich. Bitte die beiden neuen `**` vor dem `.mp3` beachten.

```
mp3gain -r /Pfad/zur/Musik/**/*.mp3
```

Ordner automatisiert

Angenommen es wird regelmäßig Musik in einem Ordner auf einem Raspberry Pi abgelegt. Dieser stellt ein SMB-Laufwerk im Netzwerk zur Verfügung, von welchem ein Musikplayer die Lieder abspielt. Dann könnte entweder jedes Mal auf dem Rechner, auf dem die Musik heruntergeladen wird, diese mit MP3Gain normalisiert werden oder auf dem Raspberry Pi wird z. B. täglich ein Skript ausgeführt.

Im folgenden Beispiel wird z. B. die Musik in den Ordner `/Musik/Uploads/` hochgeladen und dann jeden Morgen um 2 Uhr verarbeitet. Einige Musikplayer streamen die Musik im Ordner `/Musik/Archiv/`.

Dann würde folgendes Skript (`/Musik/musik_normalisieren.sh`) erstellt werden.

```
mp3gain -r /Musik/Uploads/*.mp3
mv /Musik/Uploads/*.mp3 /Musik/Archiv/
```

Nun noch den Cronjob einstellen mit `crontab -e`:

```
0 2 * * * /bin/bash /Musik/musik_normalisieren.sh
```

Fertig. Ab jetzt wird jede Nacht um 2 Uhr die Lautstärke der neuen Lieder automatisch angepasst.

Weitere Parameter

Neben den schon vorgestellten Parametern, unterstützt MP3Gain noch diverse weitere.

Einige Parameter mit Beispielen und Erklärungen können z. B. hier nachgelesen werden: [Ubuntuusers MP3Gain]

Am besten können die aktuell verfügbaren Parameter mit `mp3gain -?` abgefragt werden.

Hierdurch wird folgende Ausgabe erzeugt.

```
> mp3gain -?
mp3gain version 1.6.2
copyright(c) 2001-2009 by Glen Sawyer
uses mpglib, which can be found at http://www.mpg123.de
Usage: mp3gain [options] <infile> [<infile 2> ...]
options:
        -v - show version number
        -g <i>  - apply gain i without doing any analysis
        -l 0 <i> - apply gain i to channel 0 (left channel)
                  without doing any analysis (ONLY works for STEREO files,
                  not Joint Stereo)
        -l 1 <i> - apply gain i to channel 1 (right channel)
        -e - skip Album analysis, even if multiple files listed
        -r - apply Track gain automatically (all files set to equal loudness)
        -k - automatically lower Track/Album gain to not clip audio
        -a - apply Album gain automatically (files are all from the same
                     album: a single gain change is applied to all files, so
                     their loudness relative to each other remains unchanged,
                     but the average album loudness is normalized)
        -m <i> - modify suggested MP3 gain by integer i
        -d <n> - modify suggested dB gain by floating-point n
        -c - ignore clipping warning when applying gain
        -o - output is a database-friendly tab-delimited list
        -t - mp3gain writes modified mp3 to temp file, then deletes original
             instead of modifying bytes in original file (default)
        -T - mp3gain directly modifies mp3 file (opposite of -t)
        -q - Quiet mode: no status messages
        -p - Preserve original file timestamp
```

```
        -x - Only find max. amplitude of file
        -f - Assume input file is an MPEG 2 Layer III file
             (i.e. dont check for mis-named Layer I or Layer II files)
        -? or -h - show this message
        -s c - only check stored tag info (no other processing)
        -s d - delete stored tag info (no other processing)
        -s s - skip (ignore) stored tag info (do not read or write tags)
        -s r - force re-calculation (do not read tag info)
        -s i - use ID3v2 tag for MP3 gain info
        -s a - use APE tag for MP3 gain info (default)
        -u - undo changes made (based on stored tag info)
        -w - "wrap" gain change if gain+change > 255 or gain+change < 0
              (use "-? wrap" switch for a complete explanation)
If you specify -r and -a, only the second one will work
If you do not specify -c, the program will stop and ask before
     applying gain change to a file that might clip
```