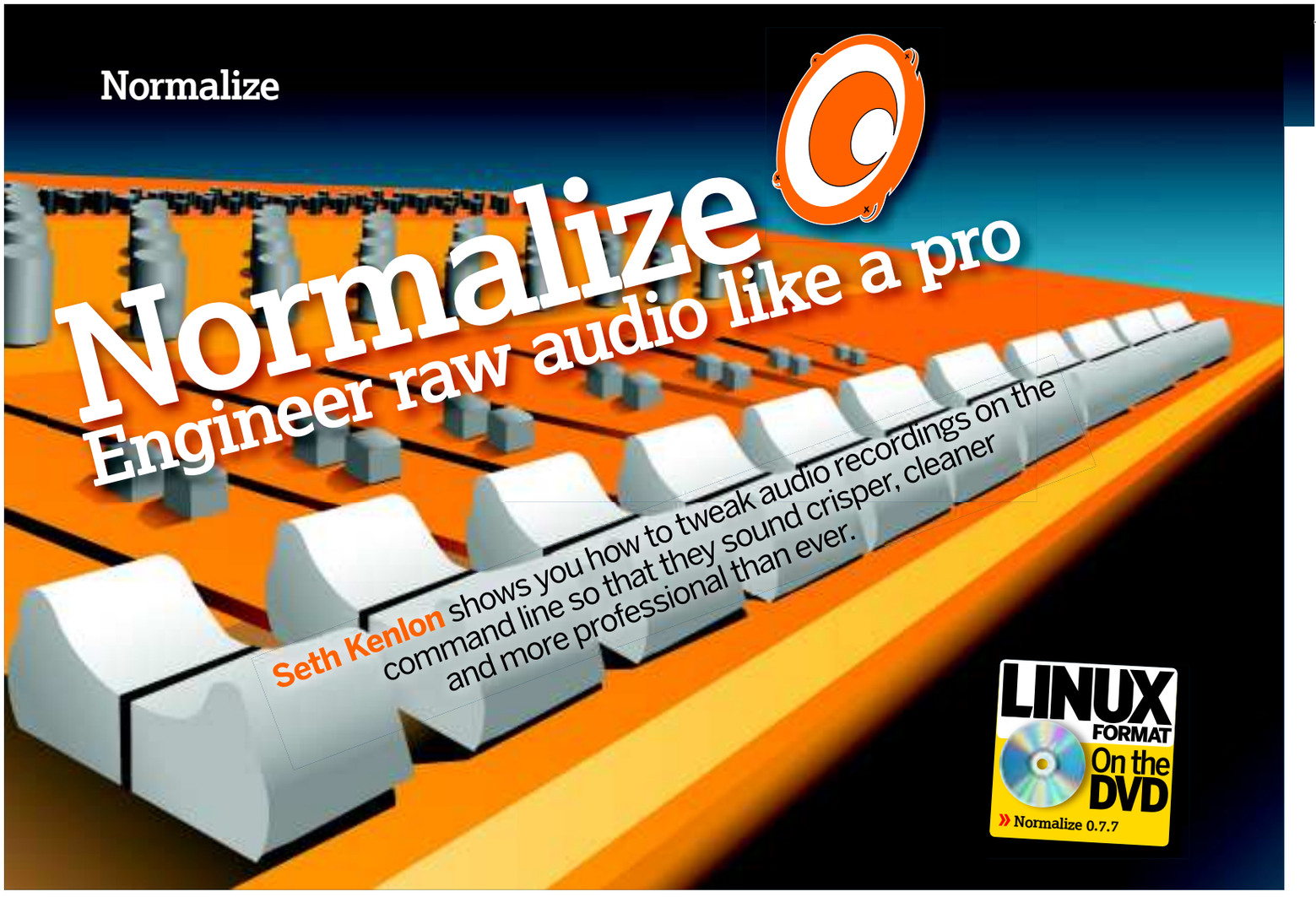


Normalize
Engineer raw audio like a pro

Seth Kenlon shows you how to tweak audio recordings on the command line so that they sound crisper, cleaner and more professional than ever.

LINUX FORMAT On the DVD
» Normalize 0.7.7

Our expert

Seth Kenlon is a video and film professional and Linux geek, out to prove that the two are not mutually exclusive. His distros of choice are Slackware and Fedora.

If you've ever tried recording audio, whether it's a brainstorming session, a weekly podcast, a talk at a technical convention or just a few random ideas spouted off quickly on to your phone, you'll know that sound can be difficult to perfect. With the help of an application called *Normalize*, we'll make your raw audio sound even and clear, and reduce the signal-to-noise ratio – all from within a command line interface.

First, a bit about audio. It's full of variation. Look at a graphical representation of sound waves and you'll see that they have peaks and troughs, big and small. Some jump up suddenly and others slope gently over a long period of time. These are all important characteristics, but when you have too much variation, at best you'll be forcing people to ride their volume control, and at worst you'll discourage them from listening to your recording at all.

Normalize is a command line application dedicated to adjusting audio to avoid wild variation in sound levels. This means you'll be able to bring all volume levels in a file to an average point, resulting in audio that's both clear and easy on the ears. It will also, by default, minimise any background noise, so that even if you don't have access to a professional recording booth, you'll be able to reduce all of those annoying sounds that made it on to your recording.

Let's take the raw audio for a podcast such as *Linux Format*'s on *TuxRadar*. Some presenters are louder than others; at times someone will whisper for effect and at other times people shout to make a point. We want to keep some variation, so that shouts come across as shouts and whispers remain whispers, but we'd also like the general speaking voices to be audible at a normal listening level.

> **"We'll make your raw audio sound even and clear."**

Volume is relative. One presenter might be shouting but if the listener simply reaches over and turns down the volume, the shout may as well be a whisper. What makes volume effective in a recording is its relation to a normal, comfortable listening level.

Understanding sound levels

Happily, the recording engineers of the world have a way of handling this. They assume that -12dB or -15dB (it's not written in stone, so you'll find some variation here and there) is the normal listening level. The audience will adjust their media players to make anything at that level comfortable for themselves. Sounds intended to be heard as loud, emphatic or explosive will be progressively above that baseline, while anything meant to be soft, quiet or subtle will be lower.

Why -15dB? Because anything above 0dB will blow the speakers, so with everything set at normal (-15dB), you have plenty of room above that to make elements of your recording different levels of loud.

Normalise song lists

When you make a mix CD using songs from various sources, tracks from one album will often have been mastered at different levels from those on another, meaning you have to adjust the volume constantly. The answer is to find the average level of each song and then individually normalise it to meet that average. The command syntax for this is: **bash$ normalize --mix song1.ogg song2.ogg song3.ogg song4.ogg**. The rest is up to *Normalize*, and it does this well.

So, in a podcast consisting mainly of people talking, the goal will be to make all of the voices as loud as possible without clipping the waveforms (and therefore producing distortion for the listeners), while keeping some of the natural fluctuation in sound levels to account for the way the human voice works. A simple command such as:

```
bash$ normalize audiofile.wav
```

is a great place to start. *Normalize* does a good job of analysing and averaging the levels it finds in an audio file. If only one is given, its waveforms are examined, an optimal average level calculated and the adjustments are made.

However, we wouldn't be sound engineers if we didn't tweak a few settings. The **--no-adjust** option will prove invaluable, as it places *Normalize* in an analyse-only mode and prevents it from altering a file. This is the **-n** flag, and we can use it to compare the results of some of the tests we perform, or to diagnose what needs to be done to an audio file before attempting any normalisation.

```
bash$ normalize -n foobar.wav
Computing levels...
   level       peak        gain
-29.3460dBFS -9.6218dBFS  17.3460dB  foobar.wav
```

This shows you the average level of the audio file (-29.34dB), where the levels peak, and the safe amount of gain you could apply to the file without clipping sound waves. It's helpful information that you'll be able to use before applying normalisation or other options to your audio.

It's also a good idea to make a copy of whatever file you're going to manipulate, as *Normalize* doesn't output to a copy, but back to the original.

Power tools

The next tool we'll wield is **-a**, a powerful switch that we can use to control the amplitude of a file. Amplitude here refers to something called RMS (root mean square), or the perceived sound level of something. Assigning the decibel value of **foo** to **--amplitude** (or **-a** for short) sets the average level of the audio file to **foo**. For instance, where **foo** is **-15dB**:

```
bash$ normalize --a -15dB foobar.wav
Computing levels...
 foobar.wav    100% done, ETA 00:00:00 (batch 100% done,
ETA 00:00:00)
Applying adjustment of 0.35dB to foobar.wav...
 foobar.wav    100% done, ETA 00:00:00 (batch 100% done,
ETA 00:00:00)
bash$ normalize -n foobar.wav
Computing levels...
   level       peak        gain
-15.6998dBFS -0.0649dBFS  3.6998dB   foobar.wav
```

It's worth mentioning that *Normalize*'s default settings are often very good, and you may never have to use the **-a** switch. However, sometimes it needs a bit of guidance, and **-a** works well for that.

There's a variation on this switch that uses the peaks for the basis of the average and it is, predictably, **-p** or **--peak**. Going hand in hand with adjusting loudness is the device that prevents sound waves that are already loud from getting even louder to the point of hitting the red (that is, going above 0dB). This device, in the hardware world, is called a Limiter and Compressor, and the software equivalent in *Normalize* is the **-l** or **--limiter** flag.

This controls the limiter by permissible peak level; set **foo** to the maximum peak you want any sound wave in your file to reach. Its default is 0dB; the loudest level possible before clipping. That's a dangerous maximum setting, because you'll often be mixing sound files together and as two tracks mix, their loudness compounds. You may find that your VU meters (in *Audacity* or *Qtractor* and so on) hit the red zone less if you set your limiter to -3dB or so.

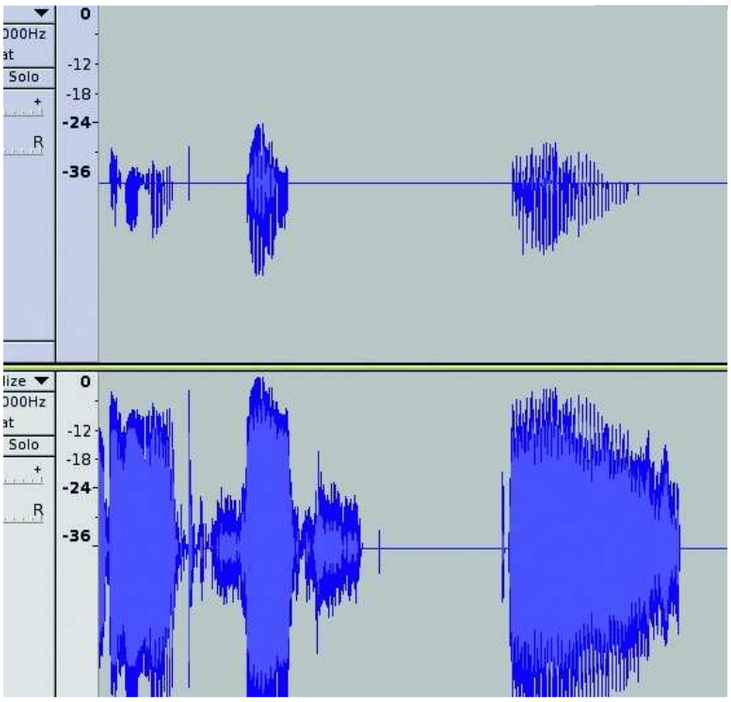
By now you might be wondering why we don't simply boost the volume levels of the sound clip and be done with it. Why not just run normalise **-n** on the file, determine how much gain can safely be applied and pump up the level accordingly? Well, technically it's possible to do this, but it's far from ideal.

No gain, no pain

Gain indiscriminately increases all sound in the file, even unwanted background noise such as outside traffic or a nearby humming server. What a good limiter does for you is create what's called a noise gate, meaning that when *Normalize* analyses the audio file, it determines what sound is lower than everything else and assumes that at this low level, it can safely be assumed there dwells only background noise. When it applies its adjustments, it doesn't apply them to this level of sound, thereby increasing the "audio contrast", or the difference between the clearly audible signal versus the now very quiet background noise.

So, using one command coupled with a few switches, we've successfully raised the volume of our audio file while keeping the background noise where it was to begin with. We've limited the sound from getting too loud, however, and have managed to maintain some of the internal fluctuation in the voices, providing a good average level overall.

From these two simple projects, the sheer power that dwells in the Linux command line becomes apparent. With tools such as *Normalize*, *Sox*, *FFmpeg*, *Mencoder* and others, audio processing can be scripted, automated or done manually without a GUI. *Normalize* provides great default presets, as well as a host of options to customise sound processing, and can help you optimise audio for all of your multimedia needs. LXF



Source and normalised audio in *Audacity*. Notice the distinct lack of background noise in the edited version, even though the signal is stronger.