FINALISING AND MASTERING AN AUDIO PROJECT

Audio and music production

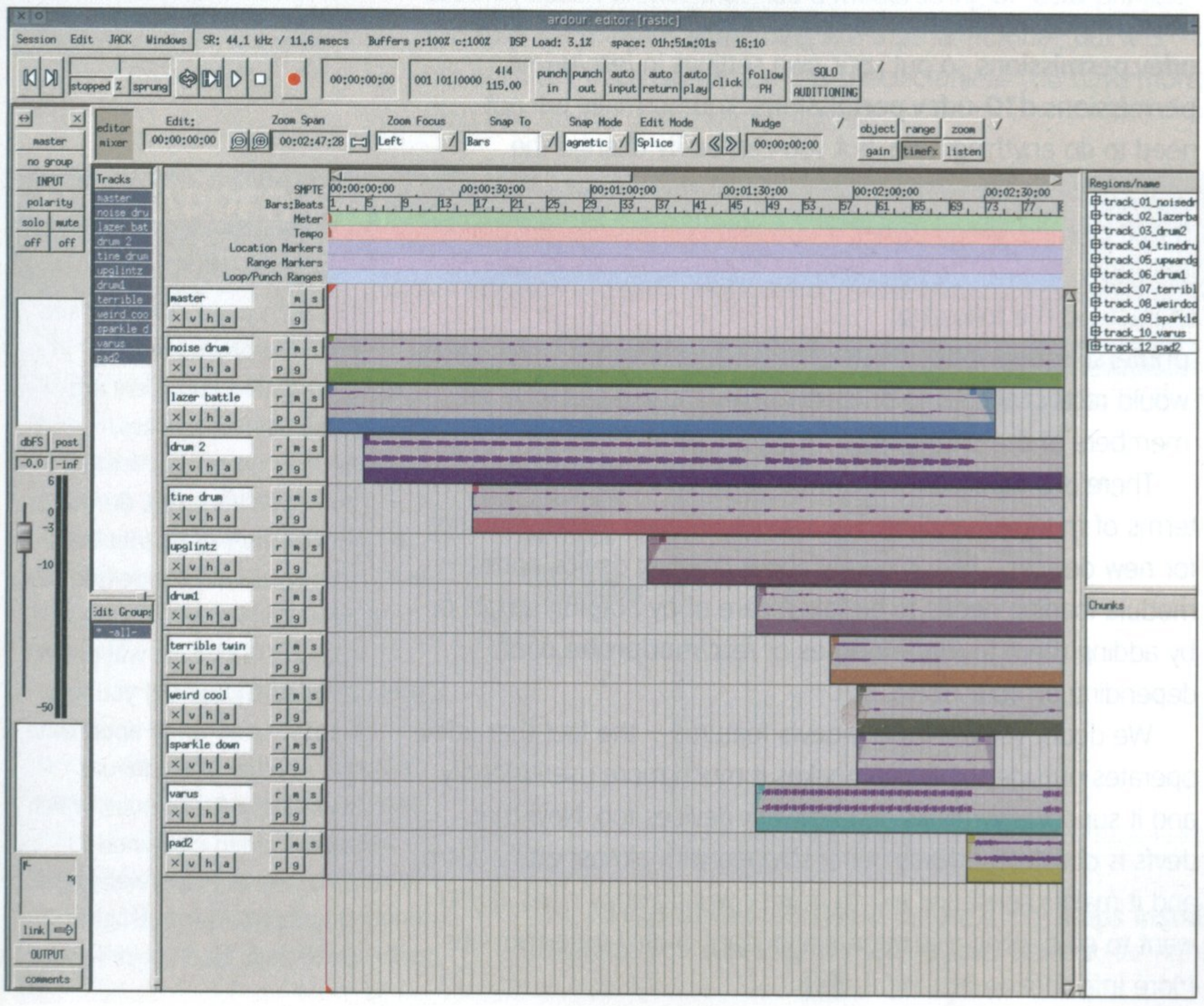
PART 4 **Mastering an audio project involves fixing all its minor inconsistencies. Graham Morrison takes to the controls for the final stage in the process.**

We're going to cover the final stages of a typical audio project in our final Linux audio tutorial. This is where the composition's *galácticos* are made into a team: frequency ranges are massaged, a little more energy is squeezed out of certain parts, while others need to be restrained. It's what's known in the industry as mastering, and is usually performed by highly-skilled engineers, as a fresh set of ears can make the difference between success and failure. This is the only way to finish a piece professionally and is becoming part of the compositional stage with many virtual studios.

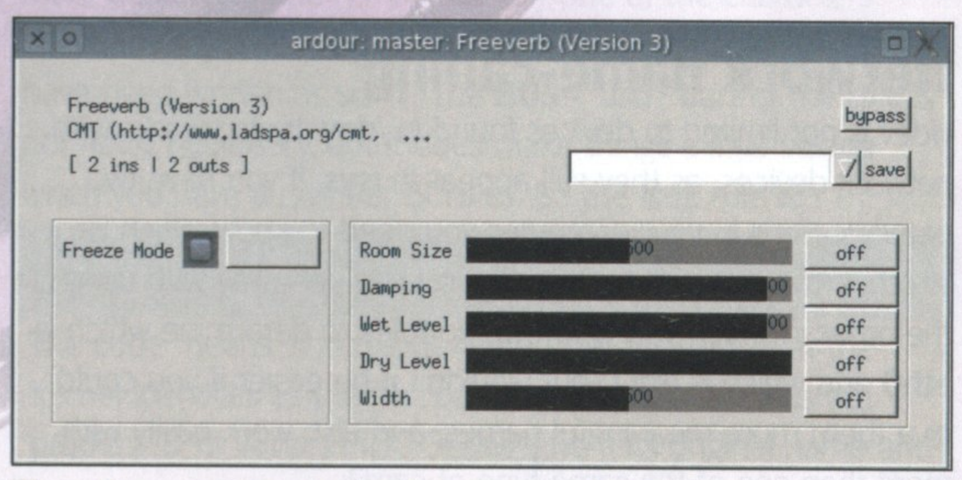
Mastering uses a just few essential audio processes. We briefly covered the basics of these effects during the first audio tutorial but it's worth covering them in more detail. There are broad divisions in the way that effects are used and implemented. Some are purely utilitarian, while others are creative, but the majority fall somewhere between. They can be classified into three categories according to how they act on the sound: delays, filters and dynamics. Obviously, the best way to get to grips and understand the various effects available is to use them, and currently, the application that provides the best environment for effects processing is *Ardour*.

Adding effects

Any Jack-compliant application can be used for inserting effects into the audio chain, but *Ardour* has been built from the ground up to take maximum advantage of this interconnectivity. This means that effects can be inserted manually into each channel, or you can use a function called a send in the same place. Each send creates a separate Jack channel where a track's audio



The final project in *Ardour*.



There's more to reverb than meets the ear.

can be sent to an external program and re-inserted back into the track after processing. This is the same as working with external processors in a studio – which you can still do, simply by making the Jack channel connect to physical input and output ports.

Ardour is also well suited to mastering a project, thanks mainly to its ability to perform the same effect-sending trick from the master bus – right at the end of *Ardour*'s audio chain. This makes it perfect for the final optimisations that often need to be made to a piece before you burn it onto a CD. To get the same functionality out of *Rosegarden* would involve connecting the mastering effects manually into the final audio stream from Jack, and *Rosegarden* has neither knowledge nor control over what happens to this stream once it leaves the application

Effects are inserted into *Ardour* on a track-by-track basis, using a working style very similar to that of *Rosegarden*. In fact, they both feature functionally-similar mixing windows, which is no surprise given that they were both designed to replace dedicated mixers. The main difference is that *Ardour*'s mixer features white space before and after a channel's fader for inserting effects. *Rosegarden* only supports effects added after the fader, and while it may not sound like it, there is a distinction between the two. As the fader controls the volume level of a track, it means that effects inserted before the fader will process the original audio level before it can be amplified or attenuated: those inserted afterwards process the audio post-filter. Depending on the effect, this can make either a little or a lot of difference, but we'll get to that.

Once audio has been loaded and dropped into its own track in *Ardour*, effects can be assigned to the track from the mixer window or the channel strip in the arrange window. This is done by right-clicking in the small box below the channel's fader (the post-effect slot). The corresponding box above the fader is for pre-fader effects. LADSPA-compatible effects can be inserted by selecting New Plugin – the first of these is a delay.

Delays, filters and dynamics

Delay effects don't just work on sound in the immediately obvious way. Many processes rely on an array of tiny delays to generate a more complex effect than that of simply playing back a section of audio at a later time. The most widely used of these in audio production is reverb (an abbreviation of reverberation). This is an attempt to create an acoustic

ON THE DVD

ON the DISC

A complete 22 track audio project to test your new skills

On this month's DVD you should find a complete audio project constructed using software from the last few tutorials. The project includes 11 stereo tracks that can be loaded into *Ardour*. Not only is this useful to see how projects can be constructed, but this particular project is perfect for practising your new-found mastering skills, as all the audio contained in the piece is quite rough and untamed (there is also an example of a mastered version of the track on the DVD).

Audio takes up a lot of space (without compression these files would take up over 800MB of space), so they've been compressed using *Ogg Vorbis*. As long as you have this already installed, they can easily be converted back to WAV files using this command from the directory containing the files:

```
$ oggdec *.ogg
```

The best way to get the converted files into *Ardour* is to create a new 16-track project by selecting New from the session menu, then entering a project location before switching to the configuration tab and selecting 16 Tracks before clicking on Create.

After you've created the project, the audio files can be imported into *Ardour* by right-clicking in the Region pane on the right-hand side of the arrange window – then you can either import a copy, or import a link to the audio. Once each segment is listed on the right it should be dragged to the beginning of a separate track. Pressing Play should let you hear the whole piece.

This project presents plenty of opportunities for adding effects to the tracks (especially the pad track which really needs something like a flange or a phaser), and is also perfectly suited to passing through a mastering suite such as *Jamin*.

This music has been released under the Creative Commons Attribution-NonCommercial-ShareAlike licence. Despite its overly complex name, it's very similar to the GPL licence familiar to application development. You can redistribute and modify the files as long as subsequent versions share the same license, but the audio can't be used commercially.

For further information on the legality and the motivation behind the Creative Commons licences see http://creativecommons.org/.

environment that differs from that of the source material. Audio in a recording environment is usually recorded dry – that is, without any external ambiance or effect. A dry signal is versatile, and doesn't impose its recording environment on other tracks.

The reason that reverb is considered a kind of delay is that it's usually nothing more than calculated reflections and filters from a mathematical model of a room. In a canyon-sized room you'd get an obvious echo, but with a smaller environment you get an almost imperceptible impression of space. As a result, these algorithms are computationally-intensive, and it's only relatively recently that native processing has been able to compete with external digital signal processing hardware. Notable reverb effects for Linux include *Freeverb* and *gverb*, and they're both perfect for augmenting a synth such as one created with *AMS*.

Other notable delay-based effects include phase and flanger. Both use discrete delays to create similar effects. The flanger was famously invented to accommodate John Lennon's hate of multi-tracking recording, using two tape machines and varying the playback speed of one by holding a finger on the machine's reel flange to produce timing fluctuations.

The main function for filters, at least at the mastering stage, is equalisation. In a similar way to those gigantic graphic equalisers on the front of an eighties hi-fi, they can either amplify or reduce the audio signal at varying frequencies.

The difference in the studio is that the frequency range can usually be configured, as can the range and density of the effect. Early digital designs were aimed at creating the perfect equaliser, but it was soon discovered that there was always something missing from their character, in the same way as with the filters for synthesizers. This has resulted in modern designs trying to incorporate some of the imperfections of those older hardware devices.

The least obvious but perhaps most important group of effects are those involved in changing the dynamics of the audio. The simplest such device is a noise gate, which mutes the sound when the level falls below a certain threshold. This is especially useful for eliminating the background noise from a typical guitar amplifier.

Another dynamic effect is called a compressor (otherwise known as 'the bane of modern music'). When misused, it forces a uniform signal level across a whole track. Music shouldn't be this way, but that doesn't mean that a compressor doesn't have its uses. In fact, it's pretty much the only way to bring out some of the detail that often lies just under the surface of a recording. This is most obvious when recording vocals: one look at the waveform would show that there's often wide variation in the level of the signal (imperceptibly compensated for when listened to in isolation).

This can make the track difficult to hear when combined with the other tracks in a project. That's where the compressor comes in. As its name suggests, it compresses the dynamic »

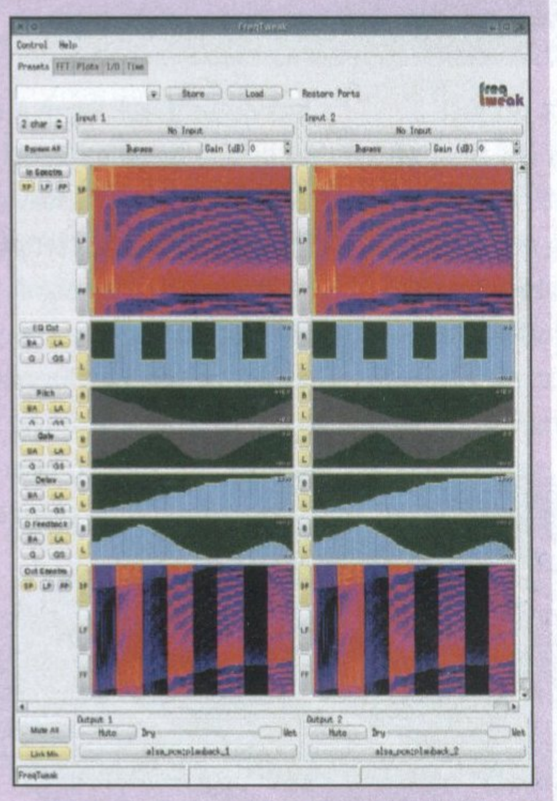
FREQUENCY EFFECTS

Multi-band mayhem with *FreqTweak*

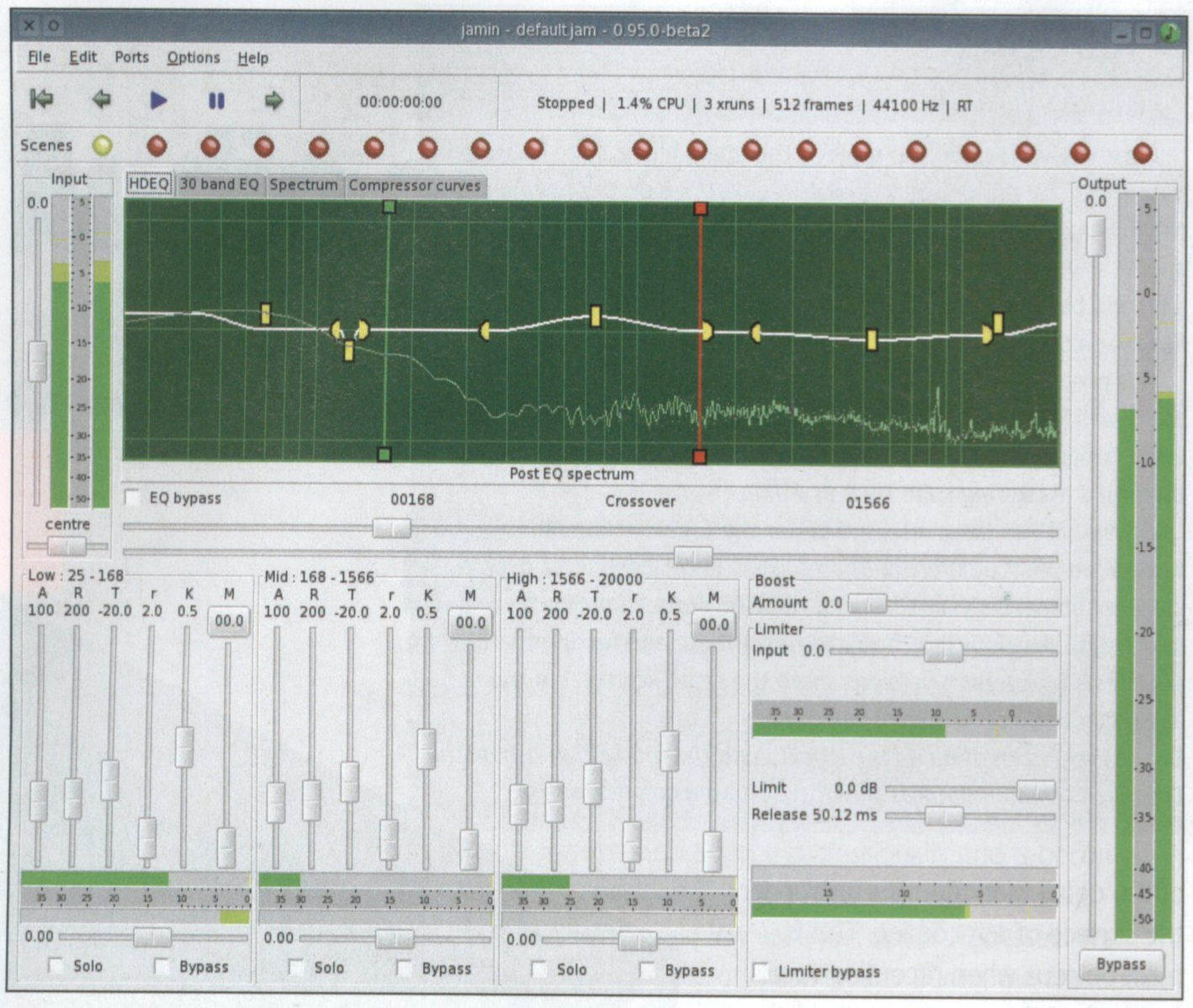
For generating a modest amount of audio mayhem, try passing your audio through *FreqTweak*, a graphical frequency-dependent filter. It also has more practical uses (you can use it to monitor audio, for example), as both the input and output audio stream can be viewed as a stereo spectrogram.

Both of the colourful graphs at the top and bottom of *FreqTweak*'s main window plot the volume level of each frequency in a horizontal spectrum of colour. Blue represents the lower levels, red hues furnish the middle and violet denotes higher frequencies. Time is shown on the vertical axis, so as the audio passes through the spectrograph, patterns of frequency look like rainbows at night.

The only difference between *FreqTweak* and a conventional effects processor is that the user specifies which frequencies are affected (you do this by drawing them on a histogram in *FreqTweak*'s GUI. The first effect, for instance, is a filter, and removes the frequencies from the audio path depending on the levels drawn into its corresponding histogram.



***FreqTweak* enables you to draw filters and delays then shows you the difference using a pair of spectrograms.**



***Jamin*'s interface isn't as bad as its name.**

range (that's the gap between the loudest and the quietest parts), generating a more consistent level. In simple terms, it attenuates the loudest parts while amplifying the quieter.

The whole point in covering these processes in some detail is that they're all involved in the mastering stage of a project. Using a compressor with equalisation on a vocal track is often essential. The same is true, to a greater or lesser extent, of the other components in a project. Bass tracks rely on compression to create the pumping punctuation in a typical dance track as much as compression on a vocal can change the emphasis in a performance, so it's not all about volume.

If you've managed to take all this in, we can now move on to the mastering stage. Linux features a specific mastering application, which in keeping with other dreadfully-named Linux audio utilities is called *Jamin*. It's basically a hard-wired group of mastering effects, arranged as you might typically use them for finalising a project. This is meant to be the last process in the audio chain before the track is burned to disc (actually the very last should be a dither algorithm for making best use of your native 32-bit sound files when they're converted to 16-bits for CD).

Master and servant

Ardour's master bus has a section for processors that need to occupy the final stage of an audio chain. It is below the master channel fader (labelled Post-Fader Inserts, Sends & Plugins). As *Jamin* is a stand-alone application rather than a plugin, Jack connections to *Jamin* need to be made from the master bus. Right-clicking in the white box below the master channel's fader should present you with the option of inserting a new plugin, a send or an insert. In this context a plugin is for internal LADSPA effects; a send simply pipes the output from the channel to an external process; and an insert does the same but expects the signal to be directed back to the channel. It's this final one that we need to use so that the output from *Jamin* comes back to *Ardour*'s master channel.

Once an insert has been created, *Jamin* itself needs to be started. By default, it automatically connects itself to the hardware outputs in Jack. This would create a problem on playback, as you'd get a duplication of the signal, one being from *Ardour*, the other from *Jamin*. To prevent this, just disconnect *Jamin*'s output from within *qjackctl*'s connection window. The next step is to wire *Jamin* into the audio chain. While this can be done from *qjackctl*'s connection window, *Ardour* has a considerably simpler interface and is often far easier to use than the spaghetti wiring you often get using *qjackctl*. To make this connection from within *Ardour*, right click on the newly-created insert and select Edit. This brings up *Ardour*'s own connection window, with the inputs on the left and the outputs on the right. At the moment, there's only a single output enabled – the equivalent of sending audio in mono. This is useful for some effects, but would be catastrophic for the master track.

To add the other channel, just click on Add Output. To make the connection, select the *Jamin* tab for both the input and output channels, followed by clicking on the *Jamin* outputs as they appear in the list. This should move them to the output box on the left of each section and at the same time make the connection within Jack. Finally, the insert needs to be enabled, either from the menu, or by middle clicking on the insert (this should remove the brackets).

If all has gone well, when you next play through the project, the output should be routed first to *Jamin* and from there back to *Ardour*. You can tell that *Jamin* is working when there's lots of activity in its window. That window is a little intimidating at first, but it's really only providing access to a combination of three compressors and a parametric equaliser, plus a couple of

TAKING CONTROL

External controllers and Linux

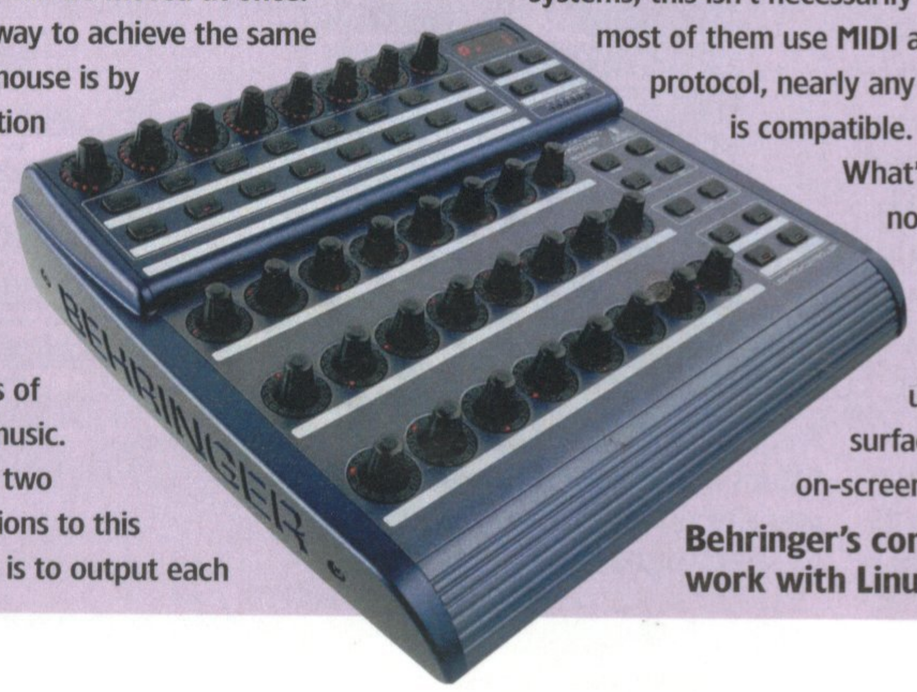
Getting the volume levels right for a project with 11 tracks of audio is difficult at the best of times, but moving virtual sliders with a mouse button makes it even harder. Mixing consoles give music engineers instant access to a track's volume through its corresponding fader, and several faders can be moved at once.

The only way to achieve the same thing with a mouse is by using automation to record the movement on each channel's fader over several passes of the piece of music.

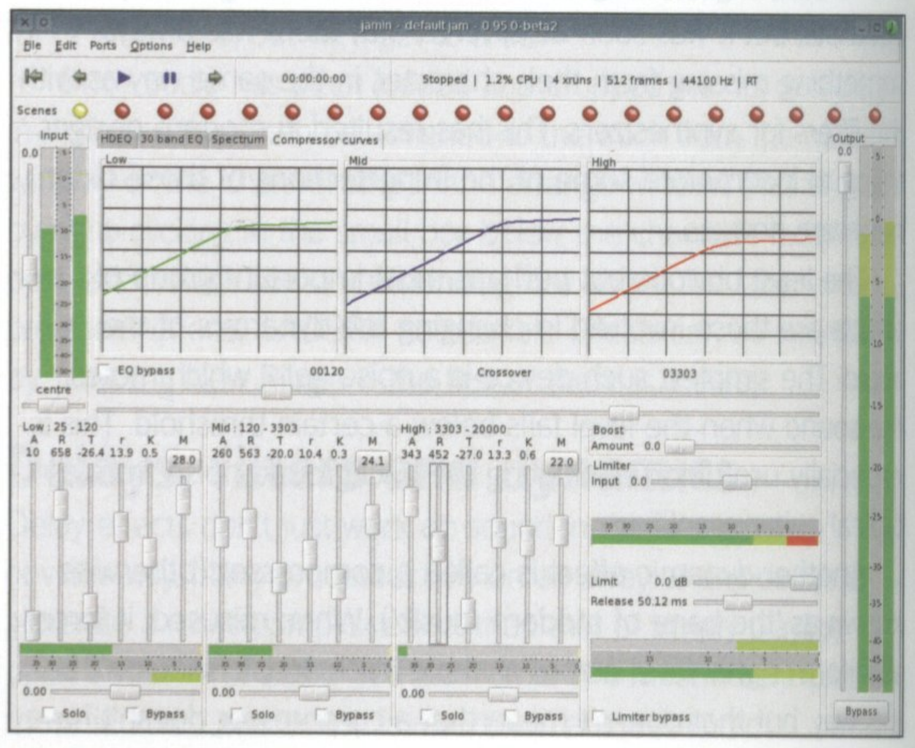
There are two possible solutions to this problem. One is to output each individual track from a well-specified hardware interface to an external mixer. The other way is to use an external hardware controller that's interfaced to the computer in some way.

While you may think that these controllers are the domain of other more popular operating systems, this isn't necessarily the case. Because most of them use MIDI as their transport protocol, nearly any generic interface is compatible.

What's more, there are now interfaces that accept incoming MIDI, which enables them to update their surfaces to reflect the on-screen version.



Behringer's controllers work with Linux.



The three compressors act on different frequencies.

boost and limit sliders. Why three compressors? Well, each is frequency dependant, which means they each process a separate frequency range. To help with this, the upper area of *Jamin*'s display houses a spectrum analyser, which maps the volume on the Y axis and the frequency on the X. Lower frequencies are on the left, higher on the right, and the frequency at the cursor position is displayed under the window.

Parametric equalisation can be edited either from the spectrum analyser or from the more familiar slider interface presented under the 30 Band EQ tab. From the spectrum analyser the equalisation curve can either be drawn by hand, or edited by dragging the curve anchor points, which are shown in yellow. Points over the middle line boost the signal, while those below reduce it: the width of the boost can be edited by broadening the curve. A tight notch at 50 or 60Hz for example could reduce hum from an electrical supply, while a broad gain at around 2–5KHz would lift a vocal slightly, whereas further gain in the 15KHz+ region can introduce an almost imperceptible definition to a recording. 15KHz is about the upper limit of human hearing – but you dog will love this effect.

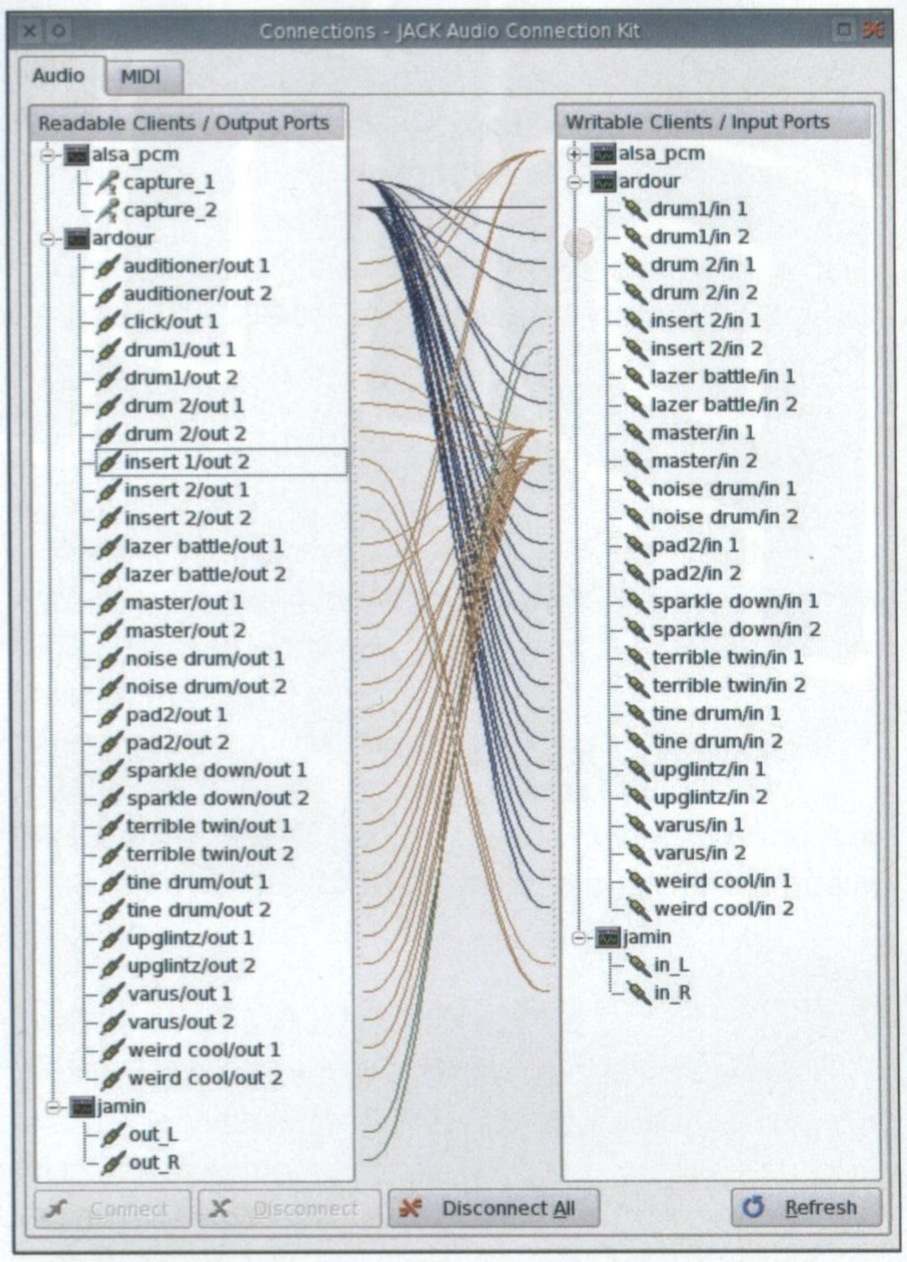
Compressed *Ardour*

The green and red vertical bands are the crossover points for the three compressors. Any frequencies that fall between the left border and the green bar are sent to the low (or bass) compressor. Those that fall between the two bars are for the mid-range, and those to the right of the red bar are sent through the high frequency compressor.

Each compressor shares the same controls, labelled as ARTrKM across the top of each section. These letters represent attack, release, threshold, ratio, knee and makeup gain, and are typical to nearly every compressor.

Beneath each compressor's sliders are a pair of level indicators. The top one displays the volume for each respective compressor's frequency, and the lower shows the gain reduction incurred on the source frequency range by each compressor. All these elements are brought together in the curves shown in the corresponding compressor tab. These are a graphical representation of the lower parameters and show the level of attenuation for any given input level. Unaltered volume would be a straight line between the lower left corner and the right-hand end of the horizontal zero gain line (values above this line represent gain).

In a typical session, you would move the threshold to the point where the gain reduction indicator starts to bounce, then alter the attack and release depending on the material and the frequency range. Slower material obviously benefits from slower envelopes, while punchier music needs the faster recovery times of a quicker envelope. It's important to hear the effect of



And you thought you could leave wires behind.

each change, and this can be made much easier using the solo switch, isolating the audio that's sent to the current compressor. The most important parameter though is ratio, as you can see when changing this value on the graph – it changes the ratio between the amount of attenuation and the input volume.

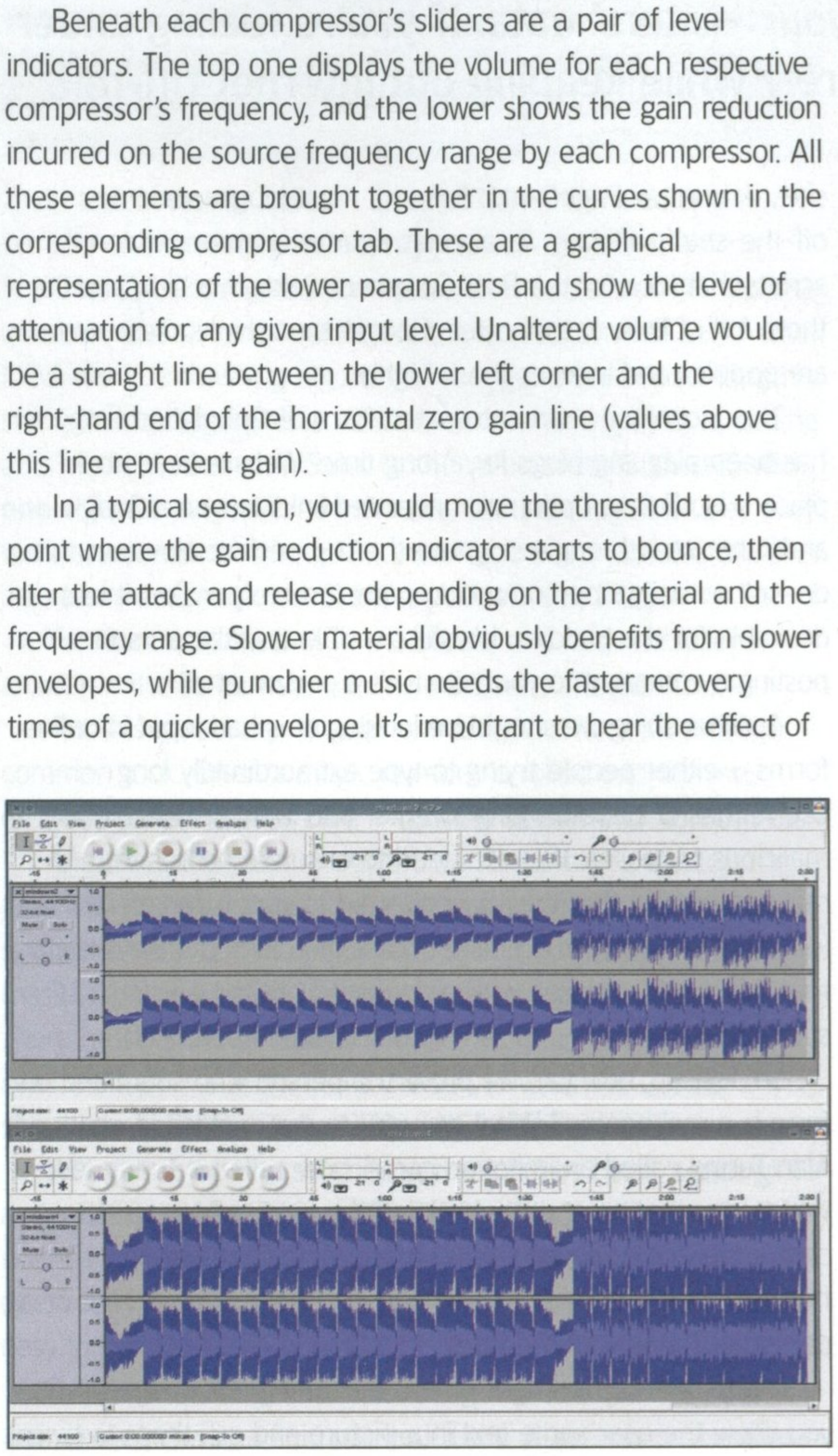
Once the project has been tweaked in *Jamin*, the next thing to do is render or record the final output to an audio file. This should make it relatively easy to either burn the track to a CD, or upload your masterpiece to a website. From *Ardour*, open the Export Session window (Session > Export > Export Session To An Audio File) and change the filename. The rest is project-dependent: for CD you would obviously need to make sure that the sample format is set to 16 bits and that the frequency is 44.1KHz.

The show must go on

If you have been working with high sample and bit rates (and as *Ardour* uses 32-bit floats for internal processing, you probably have) the output quality can be improved by using dither. Dither in audio works in exactly the same way as it does in video, and hides the imperfections inherent in digital stepping by applying a veil of noise. For best results, experiment with the different dither algorithms – but I find Shaped Noise often suits my stuff best.

Finally, make sure the master outputs are selected in the Output selection and click on Export. *Ardour* will then run through the whole project and render the output to an audio file.

And that's it. You should now have the finished product on your system in the shape of an audio file. Over the last few months, this series of tutorials has covered the basics of Linux audio, starting with the fundamentals of ALSA and Jack, followed by synthesis and sound generation before the reaching the final stages of mastering. While in many ways this has been only a brief overview of the many audio possibilities available to Linux users, the main intention has been to whet your appetite enough to push you into actually doing something, and if you do, you know where you can send the results! LXF



The before and after effects of mastering.

LINKS

Software used this month

Ardour 0.96beta26
www.ardour.org
FreqTweak 0.6.1
http://freqtweak.sourceforge.net
Jamin 0.95.0-beta2
http://jamin.sourceforge.net
SWH Plugins 0.4.13
http://plugin.org.uk

NEXT MONTH

The end is only the beginning – in next month's issue we'll look into the future with Linux and cutting-edge audio.