Linux As A Text-Based Studio
Ecasound – Recording Tool Of Choice

Julien CLAASSEN
Abtsbrede 47a
33098 Paderborn
Germany
julien@c-lab.de

Abstract

This talk could also be called "ecasound textbased harddisk recording". I am going to demonstrate a few of the most important features of ecasound and how to make good use of them in music recording and production.

This talk explains what ecasound is and what its advantages are, how a braille display works, Ecasound's basic features (playback, recording, effects and controllers), and a few of ecasound's more advanced features (real multitrack recording and playback and mastering).

Keywords

audio, console, recording, text-based

1 What is Ecasound?

1.1 Introduction to Ecasound

Ecasound is a textbased harddisk recording, effects-processing and mixing tool. Basically it can operate in two ways:

- It can work as a commandline utility. Many of its features can be used from the commandline, via a whole lot of options.
- It can also be operated from a shell-like interface. This interface accepts its own set of commands, as well as commandline options.

Ecasound supports more than your usual audio io modes:

- ALSA - Advanced Linux Sound Architecture
- Jack - Jack Audio Connection Kit
- ESD - Enlightenment Sound Daemon
- Oldstyle OSS - Open Sound System
- Arts - the Arts Sound Daemon

1.2 Advantages

1. Ecasound can easily be used in shell-scripts through its commandline options. Thus it can perform some clever processing.
2. Through its shell-interface you can access realtime controls. Via its set and get commands one can change and display controller values.
3. Because ecasound does not require an X-server and a lot of other GUI overhead, it is slim and fast. On a 700 MHz processor one can run an audio-server (JACK), a software synthesizer (fluidsynth) and ecasound with 3 or more tracks without problems.
4. Ecasound is totally accessible for blind people through its various textbased interfaces. Those interfaces provide full functionality!

1.3 Disadvantages

1. Its textbased interface is not as intuitive and easy to learn as a GUI for a sighted person .
2. Its audio routing capabilities still lack certain features known to some other big linux audio tools.
3. It does not provide much support for MIDI (only ALSA rawmidi for controlling effects and starting/stopping).

2 How I Work

I work with a braille display. A braille display can display 40 or 80 characters of a screen. In textmode this is a half or full line.

The braille display has navigation buttons, so you can move the focus over the whole screen, without moving the actual cursor. Usually the display tracks the cursor movement, which is very useful most of the time. For the rest of the time, you can deactivate tracking of the cursor.

So the best programs to use are line-oriented. Thus tools with shell-interfaces or commandline utilities are the best tools for me.

N.B.: As I heard such tools are also among the top suspects for users of speech-synthesizers.

3 Usage

This chapter will give several use cases of ecasound.

3.1 Using ecasound from the command line

As already stated ecasound can – in general – be used in two ways: From the commandline and from its interactive mode. The following examples will deal with ecasound's commandline mode.

3.1.1 Playing files from the commandline

One of the simplest uses of ecasound is playing a file from the commandline. It can look like calling any simple player – like i.e. aplay. If the ecasound configuration is adjusted correctly it looks like this:

```
ecasound myfile.wav
```

or

```
ecasound -i myfile.wav
```

The "-i" option stands for input. If you wish to specify your output explicitly and do not want to rely on the ecasoundrc configuration file, you can do it like that:

```
ecasound -i myfile.wav -o alsa,p1
```

`alsa,p1` marks the alsa output on my system-configuration running ALSA. The "-o" option means output.

3.1.2 Recording files from the commandline

It is as simple as playing files. The only thing one needs to exchange is the place of the sound-device (ALSA device) and the file (`myrecording.wav`). So if one intends to record from an ALSA device called "io1" to `myrecording.wav`, one would do it like that:

```
ecasound -i alsa,io1 -o
myrecording.wav
```

It looks just like the example from section 3.1.1 with sound-objects exchanged.

3.2 Interactive mode

Ecasound interactive mode offers a lot more realtime control over the things you mean to do like starting, stopping, skipping forward or backward etc. Thus in most cases it is more suited to the needs of a recording musician. Below there are some simple examples.

3.3 Playing a file

This method of playing a file is much closer to what one could expect of a nice player. The syntax for starting ecasound is very similar to the one from 3.1.1.

```
ecasound -c -i myfile.wav [-o
alsa,p1]
```

By pressing "h" on the ecasound shell prompt you get some basic help. For more info – when trying it at home, there is the ecasound-iam (InterActive Mode) manual page.

3.4 Interactive recording

The simplest way to record a file is almost as simple as playing a file. The only thing is you have to specify the audio-input source. Btw.: The same syntax can be used to convert files between different formats (wave-file, mp3, ogg, raw audio data...).

To do a simple interactive recording, type this:

```
ecasound -c -i alsa,io1 -o
myrecording.wav
```

Again you have the interactive capabilities of ecasound to support your efforts and extend your possibilities. Besides that, it is the same as in paragraph 3.1.2.

3.5 Effects in ecasound

Ecasound has two sources for effects: internal and external via LADSPA. In the following sections both are introduced with a few examples and explanations.

3.5.1 Internal effects

Ecasound comes with a substantial set of internal effects. There are filters, reverb, chorus, flanger, phaser, etc. All effect-options start with "e", which is good to know when looking for them in the manual pages. Here is a demo of using a simple lowpass filter on a wave-audio file:

```
ecasound -i myfile.wav -efl:1000
```

which performs a lowpass filter with a cutoff frequency of 1000Hz on the file `myfile.wav` and outputs the result to the default audio device.

3.5.2 External / LADSPA effects

Ecasound can also use LADSPA effects which makes it a very good companion in the process of producing and mastering your pieces.

There are two different ways of addressing LADSPA effects: By name or by unique ID.

3.5.3 Addressing by name

With `analyseplugin` you can determine the name of a LADSPA effect like:

```
babel:/usr/local/lib/ladspa #
  analyseplugin ./decimator_1202.so

Plugin Name: "Decimator"
Plugin Label: "decimator"
Plugin Unique ID: 1202
Maker: "Steve Harris "
Copyright: "GPL"
Must Run Real-Time: No
Has activate() Function: No
Has deativate() Function: No
Has run_adding() Function: Yes
Environment: Normal
Ports:  "Bit depth" input, control, 1 to
  24, default 24
        "Sample rate (Hz)" input, control,
  0.001*srate to 1*srate, default 1*srate
"Input" input, audio, -1 to 1
"Output" output, audio, -1 to 1
```

Thus one knows that "decimator" is the name – label – of the plugin stored in decimator_1202.so. Now you can use it like that:

```
  ecasound -i file.wav
-el:decimator,16,22050
```

which simulates the resampling of the file "file.wav" at 22.05 KHz.

3.5.4 Addressing by unique ID

`analyseplugin` not only outputs the label of a LADSPA plugin, but also its unique ID, which ecasound can also use. Mostly this way is simpler, because there is less to type and you do not have to look for upper- and lowercase letters. With the following command you can use the decimator plugin by its unique ID:

```
  ecasound -i file.wav
-eli:1202,16,22050
```

This command does the same as the one before.

Although it looks more cryptic to the naked eye, it is really shorter and (once you are used to it) much simpler to type – this is at least my personal experience.

3.5.5 Effect presets

Another powerful feature of ecasound are effect presets. Those presets are stored in a simple text-file, usually `/usr/local/share/ecasound/effect_presets`. An effect preset can consist of one or more effects in series, with constant and variable parameters. What does this mean in practice? The following illustrates the use of the metronome-effect:

```
  ecasound -c -i null -pn:metronome,120
```

This provides a simple clicktrack at 120 BPM. Internally the ecasound "metronome" effect-preset consists of a sinewave at a specific frequency, a filter – for some reason – and a pulse gate. This gate closes at a certain frequency given in BPM. Would you use all those effects on the commandline directly, you would have to type a lot. Besides getting typos, you could also choose very inconvenient settings. If you use the effect preset, everything is adjusted for you.

The standard preset file contains a good collection to start with. From simple examples for learning, to useful things like a wahwah, metronome, special filter constellations, etc...

3.5.6 Controllers

Ecasound also offers a few controllers which you can use to change effect parameters while your music is playing. The simplest controller is a two-point envelope. This envelope starts at a given start value and moves over a period of time to a certain endvalue. In practice it could look like this: A user wants to fade in a track from volume 0 to 100 over 4 seconds:

```
  ecasound -i file.wav -ea:100
-kl:1,0,100,4
```

What does the first parameter of `-kl` mean? This parameter is the same for all `-k*` – controller – options. It marks the parameter you want to change. The amplifier (`-ea`) has only one parameter: the volume. Thus the first parameter is 1. The second is the start value (0), meaning the volume should start at 0, the third value is the endvalue for the envelope: Volume should go up to 100. The last value is the time in seconds that the envelope should use to move from start to end value.

Ecasound offers more controllers than this simple one. It has a sine oscillator and generic

oscillators which can be stored in a file like effect presets. Besides that you can use MIDI controllers to do some **really** customised realtime controlling.

3.5.7 An interactive recording with realtime control

Now a short demonstration of the features presented so far: A short and simple recording with some realtime-controlled effects.

The scenario is: One synthesizer recorded with ecasound and processed by a lowpass filter which is modulated by a sinewave. This will generate a simple wahwah effect. It might look like this:

```
ecasound -c -i jack_auto,fluidsynth
-o my_file.wav -ef3:5000,0.7,1.0
-kos:1,800,5000,0.5,0
```

The `-ef3` effect is a resonant lowpass filter with these parameters: Cutoff frequency in Hz, resonance – from 0 to 1 (usually) – and gain. Values for gain should be between 0 and 1. The `-kos` controller is a simple sine oscillator with the following parameters:

1. effect-parameter – parameter of the effect to modify (first parameter of `-ef3` – the cutoff)
2. start-value – lowest value for the cutoff frequency
3. end-value – highest value for the cutoff
4. frequency in Hz – the frequency at which the cutoff should change from lowest to highest values – in this case 0.5 Hz. It takes 2 seconds.
5. iphase – initial phase of the oscillator. A sinus starts at 0 and moves upwards from there. Yet one can shift the wave to the right by supplying an iphase > 0.

4 More complex work

This chapter gives some more complex usage examples of ecasound.

4.1 Chains

4.1.1 What is a chain?

A chain is a simple connection of audio objects. A chain usually contains of:

- an audio input
- effects (optional)
- an audio output

You have already seen chains, without really knowing them because even a simple thing like:

```
ecasound -i file.wav
```

uses a chain with the default output.

To explicitly specify a chain, you need to use the -a option. The above example with an explicit chain-naming, yet still unchanged behaviour looks like that:

```
ecasound -a:my_first_chain -i
file.wav (-o alsa,p1)
```

4.1.2 What is a chain setup?

A chain setup can be seen as a map of all chains used in a session. You can perhaps imagine that you can have parallel chains – for mixing audiotracks – or even more complex structures for tedious mastering and effects processing. You can store a complete chain setup in a file. This is very useful while mastering pieces.

A simple example of an implicit chain setup includes all above examples. They have been chain setups with only one chain. To store chain setups in files you can use the interactive command cs-save-as or cs-save, if you've modified an existing **explicit** chain setup.

4.2 Playing back multiple files at once

Now the user can play back a multitrack recording before having generated the actual outputmixdown.

It could look like this:

```
ecasound -c -a:1 -i track1.wav -a:2 i
track2.wav -a:3 -i track3.wav -a:1,2,3
-o alsa,p1
```

This also demonstrates another nice simplification: One can write something like `-a:1,2,3` to say that chain 1, 2 and 3 should have something in common. In this example it could be even shorter:

```
ecasound -c -a:1 -i track1.wav -a:2
-i track2.wav -a:3 -i track3.wav -a:all
-o alsa,p1
```

This line does exactly the same as the last demo. The keyword `all` tells ecasound to apply the following options to **all** chains ever mentioned on the commandline.

4.3 Recording to a clicktrack

Now one can use chains to perform an earlier recording to a clicktrack:

```
ecasound -c -a:1,2 -i alsa,io1
-a:1 -o track1.wav -a:3 -i null
-pn:metronome,120 -a:2,3 -o alsa,p1
```

This does look confusing at first sight, but is not. There are three chains in total. Chain 1 and 2 get input from the soundcard (`alsa,p1`), chain three gets null input (null). Chain 2 (soundcard) and 3 (metronome) output to the soundcard so you hear what is happening. Chain 1 outputs to a file. Now you can use `track1.wav` as a monitor and your next track might be recorded with a line like this:

```
ecasound -c -a:1,2 -i alsa,io1
-a:1 -o track2.wav -a:3 -i null
-pn:metronome,120 -a:4 -i track1.wav
-a:2,3,4 -o alsa,p1
```

This extends the earlier example only by a chain with `track1.wav` as input and soundcard (`alsa,p1`) as output. Thus you hear the click-track – as a good guidance for accurate playing – and the first track as a monitor.

4.4 Mixing down a multitrack session

Having several tracks on harddisk, the mixdown is due. First one can take a listen to the multitrack session and then store the result to a file.

Listening to the multitrack can be achieved by issuing the following command:

```
ecasound -c -a:1 -i t1.wav -a:2 -i
t2.wav -a:3 -i t3.wav -a:all -o alsa,p1
```

Now adjusting of volumes can be managed by applying `-ea` (amplifier effect) to each track. i.e.:

```
ecasound -c -a:1 -i t1.wav -ea:220
-a:2 -i t2.wav -ea:150 -a:3 -i t3.wav
-a:180 -a:all -o alsa,p1
```

This amplifies t1.wav by 220%, t2.wav by 150% and t3.wav by 180%.

Being content with volume adjustment and possibly other effects, the only thing left is exchanging soundcard output by file-output. Meaning exchange `alsa,p1` with `my_output.wav`:

```
ecasound -c -a:1 -i t1.wav -ea:220
-a:2 -i t2.wav -ea:150 -a:3 -i t3.wav
-ea:180 -a:all -o my_output.wav
```

Now ecasound will process the files and store the mixdown to disk. The last – optional – step is to normalize the file `my_output.wav` which can be performed by `ecanormalize`:

```
ecanormalize my_output.wav
```

The normalized output file overwrites the original: So **be careful**!

5 Resume

Having in theory produced a piece ready for burning on CD or uploading to the Internet, here comes the resume. It is not the same way you would do it in a graphical environment, yet it still works fine!

For me ecasound is always the tool of choice. It is a very flexible tool. Its two general modes – commandline and interactive – combined with its chain-concept make it a powerful recording and mixing program. Because ecasound has LADSPA support and can be connected to the JACK audio server it is very simple to integrate it in a linux-audio environment. You can also use it in combination with graphical tools, if you so choose.

So for those who love text interfaces, need fast and simple solutions or those who start to learn about audio-recording, ecasound can be a tool of great value.

Besides that, ecasound is of course a very good example of what free software development can do: Produce a very up-to-date piece of fine software which is fully accessible to blind and visually impaired people. Yet still it was not written with this audience in mind. There is a fairly large crowd relying on ecasound for very different kinds of work. Though it lacks a few things that others have, it is not said that ecasound can never get there. Meanwhile there are other ways to achieve what one needs to achieve, thanks to the flexibility of ecasound and the tools you can combine/connect with it.

6 Thanks and Acknowledgements

Thanks and acknowledgements go to:

- Kai Vehmanen and the ecasound crew at http://www.eca.cx/ecasound
- Of course the ALSA crew with much work from Takashi Iwai and Jaroslav Kysela at http://www.alsa-project.org
- Richard E. Furse and companions, at www.ladspa.org, for creating LADSPA in the first place
- Steve Harris and his wonderful collection of LADSPA plugins at http://plugin.org.uk
- Paul Davis and friends at http://jackit.sf.net for jackd, our favourite realtime audio server

- http://www.fluidsynth.org, namely Josh Green, Peter Hanappe and colleagues for the soundfont-based softsynth fluidsynth
- Dave Phillips and his great collection of MIDI and audio links at http://linux-sound.org
- ZKM for hosting this conference, see the official LAC webpages at http://www.zkm.de/lac

Before thanking the great bunch of people who organised and host this event, I want to mention my own webpage at `http://ltsb.sourceforge.net`.

Great many thanks to Frank Neumann, Matthias Nagorni, Götz Dipper and ZKM for organising and hosting this conference! And many thanks and apologies to all those I forgot! Sorry to you, I didn't mean to!