Songlib: post-processing filters

written by: Song Li Buser

Revision Date: March 14, 2013

Post-processing

Once an audio file is created with the **songlib** library, you can enhance the audio via post-processing. The **songlib** package comes with a number of post-processing filters, some of which work natively with RRA files and some use *sox*. The *sox*-based filters can take RRA files as input, but convert them to WAV before *sox* processes them.

RRA-based filters

Most RRA filters accept zero, one, or two file names, plus options. If given no file names, they read from *stdin* and write to *stdout*. If given one filename, they read from the given file and write to *stdout*. If given two filenames, they read from the first and write to the second. Filters marked *ABNORMAL* do not follow this pattern.

Format filters

rra2mp3,rra2ogg,rra2flac batch convert RRA files to MP3, OGG, or FLAC – this filter requires one or more command-line arguments, all of which are the names of the RRA files to be converted; it produces similarly named files with `.mp3`, `.ogg`, or `.flac` extensions, rather than with `.rra` extensions – *ABNORMAL*

flac2rra,flac2mp3,flac2ogg batch convert FLAC files to RRA, MP3, or OGG – this filter requires one or more command-line arguments, all of which are the names of the FLAC files to be converted; it produces similarly named files with `.rra`, `.mp3`, or `.ogg` extensions, rather than with `.flac` extensions – *ABNORMAL*

ogg2rra,ogg2mp3,ogg2flac batch convert OGG files to RRA, MP3, or FLAC – this filter requires one or more command-line arguments, all of which are the names of the OGG files to be converted; it produces similarly named files with `.rra`, `.mp3`, or `.flac` extensions, rather than with `.ogg` extensions – *ABNORMAL*

mp32rra,mp32ogg,mp32flac batch convert MP3 files to RRA, OGG, or FLAC – this filter requires one or more command-line arguments, all of which are the names of the MP3 files to be converted; it produces similarly named files with `.rra`, `.ogg`, or `.flac` extensions, rather than with `.mp3` extensions – *ABNORMAL*

rra2wav,wav2rra convert from RRA to WAV and vice versa

Effects filters

Typically, effects filters are strung together in a pipeline:

```
cat track1.rra | rraamplify -a 0.8 | rrachorus | rrareverb | rrastereo -r > final1.rra
```

To see the options that a filter handles, use the `-h` option, as in:

```
rraecho -h
```

rraamplify increase or decrease the volume

rrachorus add multiple voices to the incoming audio, some pitched a little bit higher and some pitched a little bit lower

rraecho add a prominent echo to the incoming audio – unlike *soxecho* (below), which adds a single echo, *rraecho* re-echos repeatedly, each subsequent echo fainter than the previous

rrafader fade in and fade out the incoming audio

rrafastmixer mix a number of audio files, given as command line arguments, into a single audio file – the output audio is assumed to be the last filename given as an argument – if the last argument is `-`, then the output audio is sent to stdout.

rraflanger add a flanging effect to the incoming audio

rrahiss add a hiss to the incoming audio

rramono convert a multichannel audio to a single channel – a single channel from the input can be selected or all channels in the input can be merged into a single channel

rrareverb add some reverb to the incoming audio – like *rraecho*, but the echo is delayed much less

Mastering filters

These filters are useful for mastering vocals:

rrasilence reduces the volume on regions that appear to be silent - it is mainly used to remove recording hiss and background noises from vocals – try the default parameters and adjust if to much or too little signal is being removed

Sox-based filters

The *sox*-based filters take zero, one, or two command-line arguments. If no arguments are given, then the filter expects the input audio to come from stdin and the output audio to go to stdout. The default is to expect a WAV as input and produces a WAV on output:

```
cat song.wav | soxecho > songecho.wav
```

This example uses the *soxecho* filter, but the behavior is the same for all *sox*-based filters. If the `-r` option is given, then the input on stdin is RRA:

```
cat song.rra | soxecho -r > songecho.wav
```

If the `-R` option is given, then an RRA is produced on stdout:

```
cat song.wav | soxecho -R > songecho.rra
cat song.rra | soxecho -r -R > songecho.rra
```

When one command-line argument is given, it is assumed to be the name of the audio input file. Sox determines whether it is a WAV or RRA file by its extension. When two command-line arguments are given, the first is assumed to be the name of the audio input file and the output is written to a file named by the second command-line argument. Thus, the following three commands are equivalent:

```
cat song.rra | soxecho -r -R > songecho.rra
soxecho -R song.rra > songecho.rra
soxecho song.rra songecho.rra
```

Finally, there is the generic *soxfilter*, which takes a `-e` option to specify the effect and a `-o` option to pass options to the effect. For example, the *soxecho* filter devolves to the following call to *soxfilter*:

```
soxfilter -e echo -o 0.8 0.9 1000.0 0.3
```

soxecho add a prominent echo to the incoming audio

soxchorus add multiple voices to the incoming audio, some pitched a little bit higher and some pitched a little bit lower

soxreverb add some reverb to the incoming audio – like *soxecho*, but the echo is delayed much less

soxflanger add a flanging effect to the incoming audio

soxbass boost the bass of the incoming audio

soxtelephone add a telephone effect to the incoming audio – the low bass and the high treble are removed to make the audio sound like it is coming over a telephone

soxfilter the generic filter on which the above filters are based