NAME

SoX—Sound eXchange—The Swiss Army knife of audio manipulation

SYNOPSIS

sox [*global-options*] [*format-options*] *infile1*
[[*format-options*] *infile2*] ... [*format-options*] *outfile*
[*effect* [*effect-options*]] ...

play [*global-options*] [*format-options*] *infile1*
[[*format-options*] *infile2*] ... [*format-options*]
[*effect* [*effect-options*]] ...

rec [*global-options*] [*format-options*] *outfile*
[*effect* [*effect-options*]] ...

DESCRIPTION

SoX reads and writes audio files in most popular formats and can optionally apply effects to them; it can combine multiple input sources, synthesise audio, and, on many systems, act as a general purpose audio player or a multi-track audio recorder.

The entire SoX functionality is available using just the 'sox' command, however, to simplify playing and recording audio, if SoX is invoked as 'play', the output file is automatically set to be the default sound device and if invoked as 'rec', the default sound device is used as an input source.

The heart of SoX is a library called 'Sound Tools'. Those interested in extending SoX or using it in other programs should refer to the Sound Tools manual page: **libst**(3).

The overall SoX processing chain can be summarised as follows:

Input(s) → Balancing → Combiner → Effects → Output

To show how this works in practise, here are some examples of how SoX might be used. The simple:

```
sox recital.au recital.wav
```

translates an audio file in Sun AU format to a Microsoft WAV file, whilst:

```
sox recital.au −r 12000 −b −c 1 recital.wav vol 0·7 dither
```

performs the same format translation, but also changes the audio sampling rate & sample size, down-mixes to mono, and applies the **vol** and **dither** effects.

```
sox −r 8000 −u −b −c 1 voice-memo.raw voice-memo.wav
```

adds a header to a raw audio file,

```
sox slow.aiff fixed.aiff speed 1·027 rabbit -c0
```

adjusts audio speed using the most accurate **rabbit** algorithm,

```
sox short.au long.au longer.au
```

concatenates two audio files, and

```
sox −m music.mp3 voice.wav mixed.flac
```

mixes together two audio files.

```
play "The Moonbeams/Greatest/*.ogg" bass +3
```

plays a collection of audio files whilst applying a bass boosting effect,

```
play −c 4 −n −c 1 synth sin %−12 sin %−9 sin %−5 sin %−2 vol 0·7 mixer fade q 0·1 1 0·1
```

plays a synthesised 'A minor seventh' chord with a pipe-organ sound,

```
rec −c 2 test.aiff trim 0 10
```

records 10 seconds of stereo audio, and

```
rec -M take1.aiff take1-dub.aiff
```

records a new track in a multi-track recording.

Further examples are included throughout this manual; more-detailed examples can be found in the separate **soxexam**(7) manual.

File Formats

There are two types of audio file format that SoX can work with. The first is 'self-describing'; these formats include a header that completely describes the characteristics of the audio data that follows. The second type is 'headerless' (or 'raw data'); here, the audio data characteristics must be described using the SoX command line.

The following four characteristics are sufficient to describe the format of audio data such that it can be processed with SoX:

sample rate
: The sample rate in samples per second ('Hertz' or 'Hz'). For example, digital telephony traditionally uses a sample rate of 8000 Hz (8 kHz); audio Compact Discs use 44100 Hz (44·1 kHz).

sample size
: The number of bits used to store each sample. Most popular are 8-bit (one byte) and 16-bit (two bytes). (Since many now-common sound formats were invented when most computers used a 16-bit word, two bytes is often called a 'word', but since current personal computers overwhelmingly have 32-bit or 64-bit words, this usage is confusing, and is not used in the SoX documentation.)

data encoding
: The way in which each audio sample is represented (or 'encoded'). Some encodings have variants with different byte-orderings or bit-orderings; some 'compress' the audio data, i.e. the stored audio data takes up less space (i.e. disk-space or transmission band-width) than the other format parameters and the number of samples would imply. Commonly-used encoding types include floating-point, μ-law, ADPCM, signed linear, and FLAC.

channels
: The number of audio channels contained in the file. One ('mono') and two ('stereo') are widely used.

The term 'bit-rate' is sometimes used as an overall measure of an audio format and may incorporate elements of all of the above.

Most self-describing formats also allow textual 'comments' to be embedded in the file that can be used to describe the audio in some way, e.g. for music, the title, the author, etc.

One important use of audio file comments is to convey 'Replay Gain' information. SoX supports applying Replay Gain information, but not generating it. Note that by default, SoX copies input file comments to output files that support comments, so output files may contain Replay Gain information if some was present in the input file. In this case, if anything other than a simple format conversion was performed then the output file Replay Gain information is likely to be incorrect and so should be recalculated using a tool that supports this (not SoX).

Determining & Setting The File Format

There are several mechanisms available for SoX to use to determine or set the format characteristics of an audio file. Depending on the circumstances, individual characteristics may be determined or set using different mechanisms.

To determine the format of an input file, SoX will use, in order of precedence and as given or available:

1. Command-line format options.

2. The contents of the file header.
3. The filename extension.

To set the output file format, SoX will use, in order of precedence and as given or available:

1. Command-line format options.
2. The filename extension.
3. The input file format characteristics, or the closest to them that is supported by the output file type.

For all files, SoX will exit with an error if the file type cannot be determined; command-line format options may need to be added or changed to resolve the problem.

Accuracy

Many file formats that compress audio discard some of the audio signal information whilst doing so; converting to such a format then converting back again will not produce an exact copy of the original audio. This is the case for many formats used in telephony (e.g. A-law, GSM) where low signal bandwidth is more important than high audio fidelity, and for many formats used in portable music players (e.g. MP3, Vorbis) where adequate fidelity can be retained even with the large compression ratios that are needed to make portable players practical.

Formats that discard audio signal information are called 'lossy', and formats that do not, 'lossless'. The term 'quality' is used as a measure of how closely the original audio signal can be reproduced when using a lossy format.

Audio file conversion with SoX is lossless when it can be, i.e. when not using lossy compression, when not reducing the sampling rate or number of channels, and when the number of bits used in the destination format is not less than in the source format. E.g. converting from an 8-bit PCM format to a 16-bit PCM format is lossless but converting from an 8-bit PCM format to (8-bit) A-law isn't.

N.B. SoX converts all audio files to an internal uncompressed format before performing any audio processing; this means that manipulating a file that is stored in a lossy format can cause further losses in audio fidelity. E.g. with

```
sox long.mp3 short.mp3 trim 10
```

SoX first decompresses the input MP3 file, then applies the **trim** effect, and finally creates the output MP3 file by recompressing the audio—with a possible reduction in fidelity above that which occurred when the input file was created. Hence, if what is ultimately desired is lossily compressed audio, it is highly recommended to perform all audio processing using lossless file formats and then convert to the lossy format at the final stage.

N.B. Applying multiple effects with a single SoX invocation will, in general, produce more accurate results than those produced using multiple SoX invocations; hence this is also recommended.

Clipping

Clipping is distortion that occurs when an audio signal level (or 'volume') exceeds the range of the chosen representation. It is nearly always undesirable and so should usually be corrected by adjusting the volume prior to the point at which clipping occurs.

In SoX, clipping could occur, as you might expect, when using the **vol** effect to increase the audio volume, but could also occur with many other effects, when converting one format to another, and even when simply playing the audio.

Playing an audio file often involves re-sampling, and processing by analogue components that can introduce a small DC offset and/or amplification, all of which can produce distortion if the audio signal level was initially too close to the clipping point.

For these reasons, it is usual to make sure that an audio file's signal level does not exceed around 70% of the maximum (linear) range available, as this will avoid the majority of clipping problems. SoX's **stat** effect can assist in determining the signal level in an audio file; the **vol** effect can be used to prevent clipping, e.g.

sox dull.au bright.au vol −6 dB treble +6

guarantees that the treble boost will not clip.

If clipping occurs at any point during processing, then SoX will display a warning message to that effect.

Input File Combining

SoX's input combiner can combine multiple files using one of four different methods: 'concatenate', 'sequence', 'mix', or 'merge'. The default method is 'sequence' for **play**, and 'concatenate' for **rec** and **sox**.

For all methods other than 'sequence', multiple input files must have the same sampling rate; if necessary, separate SoX invocations can be used to make sampling rate adjustments prior to combining.

If the 'concatenate' combining method is selected (usually, this will be by default) then the input files must also have the same number of channels. The audio from each input will be concatenated in the order given to form the output file.

The 'sequence' combining method is selected automatically for **play**. It is similar to 'concatenate' in that the audio from each input file is sent serially to the output file, however here the output file may be closed and reopened at the corresponding transition between input files—this may be just what is needed when sending audio to an output device, but is not generally useful when the output file is a normal file.

If the 'mix' combining method is selected (with **-m**) then two or more input files must be given and will be mixed together to form the output file. The number of channels in each input file need not be the same, however, SoX will issue a warning if they are not and some channels in the output file will not contain audio from every input file. A mixed audio file cannot be un-mixed.

If the 'merge' combining method is selected (with **-M**), then two or more input files must be given and will be merged together to form the output file. The number of channels in each input file need not be the same. A merged audio file comprises all of the channels from all of the input files; un-merging is possible using multiple invocations of SoX with the **mixer** effect. For example, two mono files could be merged to form one stereo file; the first and second mono files would become the left and right channels of the stereo file.

When combining input files, SoX applies any specified effects (including, for example, the **vol** volume adjustment effect) after the audio has been combined; however, it is often useful to be able to set the volume of (i.e. 'balance') the inputs individually, before combining takes place.

For all combining methods, input file volume adjustments can be made manually using the **−v** option (below) which can be given for one or more input files; if it is given for only some of the input files then the others receive no volume adjustment. In some circumstances, automatic volume adjustments may be applied (see below).

The **−V** option (below) can be used to show the input file volume adjustments that have been selected (either manually or automatically).

There are some special considerations that need to made when mixing input files:

Unlike the other methods, 'mix' combining has the potential to cause clipping in the combiner if no balancing is performed. So here, if manual volume adjustments are not given, to ensure that clipping does not occur, SoX will automatically adjust the volume (amplitude) of each input signal by a factor of $1/n$, where n is the number of input files. If this results in audio that is too quiet or otherwise unbalanced then the input file volumes should be set manually as described above.

If mixed audio seems loud enough at some points through the audio but too quiet in others, then dynamic-range compression should be applied to correct this—see the **compand** effect.

Stopping SoX

Usually SoX will complete its processing and exit automatically, however if desired, it can be terminated by pressing the keyboard interrupt key (usually Ctrl-C). This is a natural requirement in some circumstances, e.g. when using SoX to make a recording. Note that when using SoX to play multiple files, Ctrl-C behaves slightly differently: pressing it once causes SoX to skip to the next file; pressing it twice in quick succession

causes SoX to exit.

FILENAMES

The following 'special' filenames may be used in certain circumstances in place of a normal filename on the command line:

− SoX can be used in pipeline operations by using the special filename '−' which, if used in place of an input filename, will cause SoX will read audio data from 'standard input' (stdin), and which, if used in place of the output filename, will cause SoX will send audio data to 'standard output' (stdout). Note that when using this option, the file-type (see **−t** below) must also be given.

−n This can be used in place of an input or output filename to specify that a 'null file' is to be used. Note that here, 'null file' refers to a SoX-specific mechanism and is not related to any operating-system mechanism with a similar name.

Using a null file to input audio is equivalent to using a normal audio file that contains an infinite amount of silence, and as such is not generally useful unless used with an effect that specifies a finite time length (such as **trim** or **synth**).

Using a null file to output audio amounts to discarding the audio and is useful mainly with effects that produce information about the audio instead of affecting it (such as **noiseprof** or **stat**).

The number of channels and the sampling rate associated with a null file are by default 2 and 44·1 kHz respectively, but, as with a normal file, these can be overridden if desired using command-line format options (see below).

One other use of **−n** is to use it in conjunction with **−V** to display information from the audio file header without having to read any further into the file, e.g. **sox −V *.wav −n** will display header information for each 'WAV' file in the current directory.

−e This is an alias of **−n** and is retained for backwards compatibility only.

N.B. Giving SoX an input or output filename that is the same as a SoX effect-name will not work since SoX will treat it as an effect specification. The only work-around to this is to avoid such filenames; however, this is generally not difficult since most audio filenames have a filename 'extension', whilst effect-names do not.

OPTIONS

Global Options

These options can be specified on the command line at any point before the first effect name.

−h, **−−help**
Show version number and usage information.

−−help−effect=*name*
Show usage information on the specified effect. The name **all** can be used to show usage on all effects.

−−interactive
Prompt before overwriting an existing file with the same name as that given for the output file.

N.B. Unintentionally overwriting a file is easier than you might think, for example, if you accidentally enter

sox file1 file2 effect1 effect2 ...

when what you really meant was

play file1 file2 effect1 effect2 ...

then, without this option, file2 will be overwritten. Hence, using this option is strongly recommended; a 'shell' alias, script, or batch file may be an appropriate way of permanently enabling it.

−m|**−M**|**−−combine=concatenate**|**merge**|**mix**|**sequence**
Select the input file combining method; **−m** selects 'mix', **−M** selects 'merge',

See **Input File Combining** above for a description of the different combining methods.

−−octave
Run in a mode that can be used, in conjunction with the GNU Octave program, to assist with the selection and configuration of many of the filtering effects. For the first given effect that supports the **−−octave** option, SoX will output Octave commands to plot the effect's transfer function, and then exit without actually processing any audio. E.g.

```
sox −−octave input-file −n highpass 1320 > plot.m
octave plot.m
```

−q, **−−no−show−progress**
Run in quiet mode when SoX wouldn't otherwise do so; this is the opposite of the **−S** option.

−−replay−gain=track
−−replay−gain=album
−−replay−gain=off
Select whether or not to apply replay-gain adjustment to input files. The default is **track** for **play** and **off** otherwise.

−S, **−−show−progress**
Display input file format/header information and input file(s) processing progress in terms of elapsed/remaining time and percentage complete. This option is enabled by default when using SoX to play or record audio.

−−version
Show version number and exit.

−V[*level***]**
Set verbosity. SoX prints messages to the console (stderr) according to the following verbosity levels:

0 No messages are printed at all; use the exit status to determine if an error has occurred.

1 Only error messages are printed. These are generated if SoX cannot complete the requested commands.

2 Warning messages are also printed. These are generated if SoX can complete the requested commands, but not exactly according to the requested command parameters, or if clipping occurs.

3 Descriptions of SoX's processing phases are also printed. Useful for seeing exactly how SoX is mangling your audio.

4 and above
Messages to help with debugging SoX are also printed.

By default, the verbosity level is set to 2. Each occurrence of the **−V** option increases the verbosity level by 1. Alternatively, the verbosity level can be set to an absolute number by specifying it immediately after the **−V** e.g. **−V0** sets it to 0.

Input File Options
These options apply only to input files and may precede only input filenames on the command line.

−v *volume*, **−−volume=***volume*
Adjust volume by a factor of *volume*. This is a linear (amplitude) adjustment, so a number less than 1 decreases the volume; greater than 1 increases it. If a negative number is given, then in addition to the volume adjustment, the audio signal will be inverted.

See also the **stat** effect for information on how to find the maximum volume of an audio file; this can be used to help select suitable values for this option.

See also **Input File Balancing** above.

Input & Output File Format Options

These options apply to the input or output file whose name they immediately precede on the command line and are used mainly when working with headerless file formats or when specifying a format for the output file that is different to that of the input file.

−c *channels*, **−−channels=***channels*

The number of audio channels in the audio file. This may be 1, 2, or 4; for mono, stereo, or quad audio. To cause the output file to have a different number of channels than the input file, include this option with the output file options. If the input and output file have a different number of channels then the **mixer** effect must be used. If the **mixer** effect is not specified on the command line it will be invoked internally with default parameters.

−−comment *text*

Specify the comment text to store in the output file header (where applicable).

SoX will provide a default comment if this option (or **−−comment−file**) is not given; to specify that no comment should be stored in the output file, use **−−comment ""** or **−−comment=**.

−−comment−file *filename*

Specify a file containing the comment text to store in the output file header (where applicable).

−r *rate*, **−−rate=***rate*

Gives the sample rate in Hz of the file. To cause the output file to have a different sample rate than the input file, include this option with the output file format options.

If the input and output files have different rates then a sample rate change effect must be run. Since SoX has multiple rate changing effects, the user can specify which to use as an effect. If no rate change effect is specified then a default one will be chosen.

−t *file-type*, **−−type=***file-type*

Gives the type of the audio file. This is useful when the file extension is non-standard or when the type can not be determined by looking at the header of the file.

The **−t** option can also be used to override the type implied by an input filename extension, but if overriding with a type that has a header, SoX will exit with an appropriate error message if such a header is not actually present.

See **FILE TYPES** below for a list of supported file types.

−L, **−−endian=little**
−B, **−−endian=big**
−x, **−−endian=swap**

These options specify whether the byte-order of the audio data is, respectively, ‘little endian’, ‘big endian’, or the opposite to that of the system on which SoX is being used. Endianness applies only to data encoded as signed or unsigned integers of 16 or more bits. It is often necessary to specify one of these options for headerless files, and sometimes necessary for (otherwise) self-describing files. A given endian-setting option may be ignored for an input file whose header contains a specific endianness identifier, or for an output file that is actually an audio device.

N.B. Unlike normal format characteristics, the endianness (byte, nibble, & bit ordering) of the input file is not automatically used for the output file; so, for example, when the following is run on a little-endian system:

sox -B audio.uw trimmed.uw trim 2

trimmed.uw will be created as little-endian;

sox -B audio.uw -B trimmed.uw trim 2

must be used to preserve big-endianness in the output file.

The **-V** option can be used to check the selected orderings.

−N, **−−reverse−nibbles**
Specifies that the nibble ordering (i.e. the 2 halves of a byte) of the samples should be reversed; sometimes useful with ADPCM-based formats.

N.B. See also N.B. in section on **-x** above.

−X, **−−reverse−bits**
Specifies that the bit ordering of the samples should be reversed; sometimes useful with a few (mostly headerless) formats.

N.B. See also N.B. in section on **-x** above.

−s/**−u**/**−U**/**−A**/**−a**/**−i**/**−g**/**−f**
The audio data encoding is signed linear (2's complement), unsigned linear, μ-law (logarithmic), A-law (logarithmic), ADPCM, IMA-ADPCM, GSM, or floating-point.

μ-law (or mu-law) and A-law are the U.S. and international standards for logarithmic telephone audio compression. When uncompressed μ-law has roughly the precision of 14-bit PCM audio and A-law has roughly the precision of 13-bit PCM audio.

A-law and μ-law are sometimes encoded using reversed bit-ordering (i.e. MSB becomes LSB). Internally, SoX understands how to work with these encodings but there is currently no command line option to specify them. If you need this support then you can use the pseudo file types of '.la' and '.lu' to inform SoX of the encoding. See supported file types for more information.

ADPCM is a form of audio compression that has a good compromise between good audio quality and fast encoding/decoding time. It is used for telephone audio compression and places were full fidelity is not as important. When uncompressed it has roughly the precision of 16-bit PCM audio. Popular version of ADPCM include G.726, MS ADPCM, and IMA ADPCM. The **−a** flag has different meanings in different file handlers. In **.wav** files it represents MS ADPCM files, in all others it means G.726 ADPCM. IMA ADPCM is a specific form of ADPCM compression, slightly simpler and slightly lower fidelity than Microsoft's flavor of ADPCM. IMA ADPCM is also called DVI ADPCM.

GSM is currently used for the vast majority of the world's digital wireless telephone calls. It utilises several audio formats with different bit-rates and associated speech quality. SoX has support for GSM's original 13kbps 'Full Rate' audio format. It is usually CPU intensive to work with GSM audio.

−1/**−2**/**−3**/**−4**/**−8**
The sample datum size is 1, 2, 3, 4, or 8 bytes; i.e. 8, 16, 24, 32, or 64 bits.

The flags
−b/**−w**/**−l**/**−d** which are respectively aliases for **−1**/**−2**/**−4**/**−8**, and abbreviate byte, word, long word, double long (long long) word, are retained for backwards compatibility only.

Output File Format Options
These options apply only to the output file and may precede only the output filename on the command line.

−C *compression-factor*, **−−compression=***compression-factor*
The compression factor for variably compressing output file formats. If this option is not given, then a default compression factor will apply. The compression factor is interpreted differently for different compressing file formats. See the description of the file formats that use this option for more information.

FILE TYPES

File types can be set by the filename extension or the **-t** option (see above). File types that can be determined by a filename extension are listed with their names preceded by a dot. File types that require optional libsndfile support are marked '**(libsndfile)**'. File types that can be handled by libsndfile using **−t sndfile** are marked '**(also with −t sndfile)**'. This might be useful if you have a file that doesn't work with SoX's default format readers and writers, and there's a libsndfile reader and writer for that format.

.raw (also with −t sndfile)
Raw (headerless) audio files. The sample rate, sample size, and data encoding must be given using command-line format options; the number of channels defaults to 1.

.ub, **.sb**, **.uw**, **.sw**, **.ul**, **.al**, **.lu**, **.la**, **.sl (also with −t sndfile)**
These filename extensions serve as shorthand for identifying the format of headerless audio files. Thus, **ub**, **sb**, **uw**, **sw**, **ul**, **al**, **lu**, **la** and **sl** indicate a file with a single audio channel, sample rate of 8000 Hz, and samples encoded as 'unsigned byte', 'signed byte', 'unsigned word', 'signed word', 'μ-law' (byte), 'A-law' (byte), inverse bit order 'μ-law', inverse bit order 'A-law', or 'signed long' respectively. Command-line format options can also be given to modify the selected format if it does not provide an exact match for a particular file.

Headerless audio files on a SPARC computer are likely to be of format **ul**; on a Mac, they're likely to be **ub** but with a sample rate of 11025 or 22050 Hz.

.8svx (also with −t sndfile)
Amiga 8SVX musical instrument description format.

.aiff, **.aif (also with −t sndfile)**
AIFF files used on Apple IIc/IIgs and SGI. Note: the AIFF format supports only one SSND chunk. It does not support multiple audio chunks, or the 8SVX musical instrument description format. AIFF files are multimedia archives and can have multiple audio and picture chunks. You may need a separate archiver to work with them.

.aiffc, **.aifc (also with −t sndfile)**
AIFF-C (not compressed, linear), defined in DAVIC 1.4 Part 9 Annex B. This format is referred from ARIB STD-B24, which is specified for Japanese data broadcasting. Any private chunks are not supported.

Note: The input file is currently processed as .aiff.

alsa ALSA default device driver. This is a pseudo-file type and can be optionally compiled into SoX. Run **sox −h** to see if you have support for this file type. When this driver is used it allows you to open up a ALSA device and configure it to use the same data format as passed in to SoX. It works for both playing and recording audio files. When playing audio files it attempts to set up the ALSA driver to use the same format as the input file. It is suggested to always override the output values to use the highest quality format your ALSA system can handle. Example: **sox infile −t alsa default**

.au, **.snd (also with −t sndfile)**
Sun Microsystems AU files. There are many types of AU file; DEC has invented its own with a different magic number and byte order. SoX can read these files but will not write them. Some .au files are known to have invalid AU headers; these are probably original Sun μ-law 8000 Hz files and can be dealt with using the **.ul** format (see below).

It is possible to override AU file header information with the **−r** and **−c** options, in which case SoX will issue a warning to that effect.

auto This format type name exists for backwards compatibility only. If given for an input file it will be silently ignored, if given for an output file it will cause SoX to exit with an error.

.avr Audio Visual Research. The AVR format is produced by a number of commercial packages on the Mac.

.caf (libsndfile)
Core Audio File format.

.cdda, **.cdr**
'Red Book' Compact Disc Digital Audio. CDDA has two audio channels formatted as 16-bit signed integers at a sample rate of 44·1 kHz. The number of (stereo) samples in each CDDA track is always a multiple of 588 which is why it needs its own handler.

.cvsd, **.cvs**
Continuously Variable Slope Delta modulation. A headerless format used to compress speech audio for applications such as voice mail. This format is sometimes used with bit-reversed samplesthe **-X** format option can be used to set the bit-order.

.dat Text Data files. These files contain a textual representation of the sample data. There is one line at the beginning that contains the sample rate. Subsequent lines contain two numeric data items: the time since the beginning of the first sample and the sample value. Values are normalized so that the maximum and minimum are 1 and −1. This file format can be used to create data files for external programs such as FFT analysers or graph routines. SoX can also convert a file in this format back into one of the other file formats.

.dvms, **.vms**
Used to compress speech audio for applications such as voice mail. A self-describing variant of **cvsd**.

.fap (libsndfile)
See **.paf**.

.flac (also with −t sndfile)
Free Lossless Audio CODEC compressed audio. FLAC is an open, patent-free CODEC designed for compressing music. It is similar to MP3 and Ogg Vorbis, but lossless, meaning that audio is compressed in FLAC without any loss in quality.

SoX can decode native FLAC files (.flac) but not Ogg FLAC files (.ogg). [But see **.ogg** below for information relating to support for Ogg Vorbis files.]

SoX has basic support for writing FLAC files: it can encode to native FLAC using compression levels 0 to 8. 8 is the default compression level and gives the best (but slowest) compression; 0 gives the least (but fastest) compression. The compression level can be selected using the **−C** option (see above) with a whole number from 0 to 8.

FLAC support in SoX is optional and requires optional FLAC libraries. To see if there is support for FLAC run **sox −h** and look for it under the list of supported file formats as 'flac'.

.fssd An alias for the **.ub** format.

.gsm (also with −t sndfile)
GSM 06.10 Lossy Speech Compression. A lossy format for compressing speech which is used in the Global Standard for Mobile telecommunications (GSM). It's good for its purpose, shrinking audio data size, but it will introduce lots of noise when a given audio signal is encoded and decoded multiple times. This format is used by some voice mail applications. It is rather CPU intensive.

GSM in SoX is optional and requires access to an external GSM library. To see if there is support for GSM run **sox −h** and look for it under the list of supported file formats.

.hcom Macintosh HCOM files. These are (apparently) Mac FSSD files with some variant of Huffman compression. The Macintosh has wacky file formats and this format handler apparently doesn't handle all the ones it should. Mac users will need their usual arsenal of file converters to deal with an HCOM file on other systems.

ircam (also with −t sndfile)
Another name for **.sf**.

.ima (also with −t sndfile)
A headerless file of IMA ADPCM audio data. IMA ADPCM claims 16-bit precision packed into only 4 bits, but in fact sounds no better than **.vox**.

.mat, **.mat4**, **.mat5 (libsndfile)**
Matlab 4.2/5.0 (respectively GNU Octave 2.0/2.1) format (.mat is the same as .mat4).

.maud An IFF-conforming audio file type, registered by MS MacroSystem Computer GmbH, published along with the 'Toccata' sound-card on the Amiga. Allows 8bit linear, 16bit linear, A-Law, μ-law in mono and stereo.

.mp3, **.mp2**
MP3 compressed audio. MP3 (MPEG Layer 3) is part of the MPEG standards for audio and video compression. It is a lossy compression format that achieves good compression rates with little quality loss. See also **Ogg Vorbis** for a similar format.

MP3 support in SoX is optional and requires access to either or both the external libmad and libmp3lame libraries. To see if there is support for Mp3 run **sox −h** and look for it under the list of supported file formats as 'mp3'.

.nist (also with −t sndfile)
See **.sph**.

.ogg, **.vorbis**
Ogg Vorbis compressed audio. Ogg Vorbis is a open, patent-free CODEC designed for compressing music and streaming audio. It is a lossy compression format (similar to MP3, VQF & AAC) that achieves good compression rates with a minimum amount of quality loss. See also **MP3** for a similar format.

SoX can decode all types of Ogg Vorbis files, and can encode at different compression levels/qualities given as a number from −1 (highest compression/lowest quality) to 10 (lowest compression, highest quality). By default the encoding quality level is 3 (which gives an encoded rate of approx. 112kbps), but this can be changed using the **−C** option (see above) with a number from −1 to 10; fractional numbers (e.g. 3·6) are also allowed.

Decoding is somewhat CPU intensive and encoding is very CPU intensive.

Ogg Vorbis in SoX is optional and requires access to external Ogg Vorbis libraries. To see if there is support for Ogg Vorbis run **sox −h** and look for it under the list of supported file formats as 'vorbis'.

ossdsp OSS /dev/dsp device driver. This is a pseudo-file that can be optionally compiled into SoX. Run **sox −h** to see if it is supported. When this driver is used it allows you to play and record sounds on supported systems. When playing audio files it attempts to set up the OSS driver to use the same format as the input file. It is suggested to always override the output values to use the highest quality format your OSS system can handle. Example: **sox infile −t ossdsp −w −s /dev/dsp**

.paf, **.fap (libsndfile)**
Ensoniq PARIS file format (big and little-endian respectively).

.prc Psion Record. Used in some Psion devices for System alarms and recordings made by the built-in Record application. This format is newer then the .wve format that is also used in some Psion devices.

.pvf (libsndfile)
Portable Voice Format.

.sd2 (libsndfile)
Sound Designer 2 format.

.sds (libsndfile)
MIDI Sample Dump Standard.

.sf (also with −t sndfile)
IRCAM SDIF (Institut de Recherche et Coordination Acoustique/Musique Sound Description Interchange Format). Used by academic music software such as the CSound package, and the MixView sound sample editor.

.sph, **.nist (also with −t sndfile)**
SPHERE (SPeech HEader Resources) is a file format defined by NIST (National Institute of Standards and Technology) and is used with speech audio. SoX can read these files when they contain μ-law and PCM data. It will ignore any header information that says the data is compressed using *shorten* compression and will treat the data as either μ-law or PCM. This will allow SoX and the command line *shorten* program to be run together using pipes to encompasses the data and then pass the result to SoX for processing.

.smp Turtle Beach SampleVision files. SMP files are for use with the PC-DOS package SampleVision by Turtle Beach Softworks. This package is for communication to several MIDI samplers. All sample rates are supported by the package, although not all are supported by the samplers themselves. Currently loop points are ignored.

.snd See **.au**.

sndfile This is a pseudo-type that forces libsndfile to be used, even for file types normally handled internally by SoX. For writing files, the actual file type is then taken from the output file name; for reading them, it is deduced from the file and any other format parameters. This pseudo-type depends on SoX having been built with optional libsndfile support.

.sndt SoundTool files. This is an older DOS file format.

.sou An alias for the **.ub** format.

sunau Sun /dev/audio device driver. This is a pseudo-file type and can be optionally compiled into SoX. Run **sox −h** to see if you have support for this file type. When this driver is used it allows you to open up a Sun /dev/audio file and configure it to use the same data type as passed in to SoX. It works for both playing and recording audio files. When playing audio files it attempts to set up the audio driver to use the same format as the input file. It is suggested to always override the output values to use the highest quality format your hardware can handle. Example: **sox infile −t sunau −w −s /dev/audio** or **sox infile −t sunau −U −c 1 /dev/audio** for older sun equipment.

.txw Yamaha TX-16W sampler. A file format from a Yamaha sampling keyboard which wrote IBM-PC format 3·5" floppies. Handles reading of files which do not have the sample rate field set to one of the expected by looking at some other bytes in the attack/loop length fields, and defaulting to 33 kHz if the sample rate is still unknown.

.vms See **.dvms**.

.voc (also with −t sndfile)
Sound Blaster VOC files. VOC files are multi-part and contain silence parts, looping, and different sample rates for different chunks. On input, the silence parts are filled out, loops are rejected, and sample data with a new sample rate is rejected. Silence with a different sample rate is generated appropriately. On output, silence is not detected, nor are impossible sample rates. Note, this version now supports playing VOC files with multiple blocks and supports playing files containing μ-law and A-law samples.

.vorbis See **.ogg**.

.vox (also with −t sndfile)
A headerless file of Dialogic/OKI ADPCM audio data commonly comes with the extension .vox. This ADPCM data has 12-bit precision packed into only 4-bits.

.w64 (libsndfile)
Sonic Foundry's 64-bit RIFF/WAV format.

.wav (also with −t sndfile)
Microsoft .WAV RIFF files. This is the native audio file format of Windows, and widely used for uncompressed audio.

Normally **.wav** files have all formatting information in their headers, and so do not need any format options specified for an input file. If any are, they will override the file header, and you will be warned to this effect. You had better know what you are doing! Output format options will

cause a format conversion, and the **.wav** will written appropriately.

SoX currently can read PCM, μ-law, A-law, MS ADPCM, and IMA (or DVI) ADPCM. It can write all of these formats including the ADPCM encoding. Big endian versions of RIFF files, called RIFX, can also be read and written. To write a RIFX file, use the **−B** option with the output file options.

.wve Psion 8-bit A-law. Used on older Psion PDAs.

.xa Maxis XA files. These are 16-bit ADPCM audio files used by Maxis games. Writing .xa files is currently not supported, although adding write support should not be very difficult.

.xi (libsndfile)
Fasttracker 2 Extended Instrument format.

EFFECTS

Multiple effects may be applied to the audio by specifying them one after another at the end of the command line.

Note: Brackets [ ] are used to denote parameters that are optional, braces { } to denote those that are both optional and repeatable, and angle brackets < > to denote those that are repeatable but not optional.

allpass *frequency width*[**h**|**o**|**q**]
Apply a two-pole all-pass filter with central frequency (in Hz) *frequency*, and filter-width *width*: in Hz (the default, or if appended with '**h**'), in octaves (if appended with '**o**'), or as a Q-factor (if appended with '**q**'). An all-pass filter changes the audio's frequency to phase relationship without changing its frequency to amplitude relationship. The filter is described in detail in [1].

This effect supports the **−−octave** global option.

band [**−n**] *center* [width[**h**|**o**|**q**]]
Apply a band-pass filter. The frequency response drops logarithmically around the *center* frequency. The *width* in Hz (the default, or if appended with '**h**'), in octaves (if appended with '**o**'), or as a Q-factor (if appended with '**q**'), gives the slope of the drop. The frequencies at *center* + *width* and *center* − *width* will be half of their original amplitudes. **band** defaults to a mode oriented to pitched audio, i.e. voice, singing, or instrumental music. The **−n** (for noise) option uses the alternate mode for un-pitched audio (e.g. percussion). **Warning: −n** introduces a power-gain of about 11dB in the filter, so beware of output clipping. **band** introduces noise in the shape of the filter, i.e. peaking at the *center* frequency and settling around it.

This effect supports the **−−octave** global option.

See also **filter** for a bandpass filter with steeper shoulders.

bandpass|**bandreject** [**−c**] *frequency width*[**h**|**o**|**q**]
Apply a two-pole Butterworth band-pass or band-reject filter with central frequency (in Hz) *frequency*, and (3dB-point) band-width *width*: in Hz (the default, or if appended with '**h**'), in octaves (if appended with '**o**'), or as a Q-factor (if appended with '**q**'). The **−c** option applies only to **bandpass** and selects a constant skirt gain (peak gain = Q) instead of the default: constant 0dB peak gain. The filters roll off at 6dB per octave (20dB per decade) and are described in detail in [1].

These effects support the **−−octave** global option.

See also **filter** for a bandpass filter with steeper shoulders.

bandreject *frequency width*[**h**|**o**|**q**]
Apply a band-reject filter. See the description of the **bandpass** effect for details.

bass|**treble** *gain* [*frequency* [*width*[**s**|**h**|**o**|**q**]]]
Boost or cut the bass (lower) or treble (upper) frequencies of the audio using a two-pole shelving filter with a response similar to that of a standard hi-fi's (Baxandall) tone-controls. This is also known as shelving equalisation (EQ).

gain gives the dB gain at 0 Hz (for **bass**), or whichever is the lower of ~22 kHz and the Nyquist frequency (for **treble**). Its useful range is about −20 (for a large cut) to +20 (for a large boost). Beware of **Clipping** when using a positive *gain*.

If desired, the filter can be fine-tuned using the following optional parameters:

frequency sets the filter's central frequency and so can be used to extend or reduce the frequency range to be boosted or cut. The default value is 100 Hz (for **bass**) or 3 kHz (for **treble**).

width determines how steep the filter's shelf transition is and can be expressed as: a 'slope' (the default, or if appended with '**s**'), a Q-factor (if appended with '**q**'), the transition width in octaves (if appended with '**o**'), or the transition width in Hz (if appended with '**h**'). The useful range of 'slope' is about 0·3, for a gentle slope, to 1 (the maximum), for a steep slope; the default value is 0·5.

The filters are described in detail in [1].

These effects support the **−−octave** global option.

See also **equalizer** for a peaking equalisation effect.

chorus *gain-in gain-out* <*delay decay speed depth* **−s**|**−t**>
Add a chorus effect to the audio. Each four-tuple delay/decay/speed/depth gives the delay in milliseconds and the decay (relative to gain-in) with a modulation speed in Hz using depth in milliseconds. The modulation is either sinusoidal (**−s**) or triangular (**−t**). Gain-out is the volume of the output.

compand *attack1*,*decay1*{,*attack2*,*decay2*}
in-dB1,*out-dB1*{,*in-dB2*,*out-dB2*}
[*gain* [*initial-volume* [*delay*]]]

Compand (compress or expand) the dynamic range of the audio. The attack and decay time specify the integration time over which the absolute value of the input signal is integrated to determine its volume; attacks refer to increases in volume and decays refer to decreases. Where more than one pair of attack/decay parameters are specified, each channel is treated separately and the number of pairs must agree with the number of input channels. The second parameter is a list of points on the compander's transfer function specified in dB relative to the maximum possible signal amplitude. The input values must be in a strictly increasing order but the transfer function does not have to be monotonically rising. The special value **−inf** may be used to indicate that the input volume should be associated output volume. The points **−inf,−inf** and **0,0** are assumed; the latter may be overridden, but the former may not.

The third (optional) parameter is a post-processing gain in dB which is applied after the compression has taken place; the fourth (optional) parameter is an initial volume to be assumed for each channel when the effect starts. This permits the user to supply a nominal level initially, so that, for example, a very large gain is not applied to initial signal levels before the companding action has begun to operate: it is quite probable that in such an event, the output would be severely clipped while the compander gain properly adjusts itself.

The fifth (optional) parameter is a delay in seconds. The input signal is analysed immediately to control the compander, but it is delayed before being fed to the volume adjuster. Specifying a delay approximately equal to the attack/decay times allows the compander to effectively operate in a 'predictive' rather than a reactive mode.

See also **mcompand** for a multiple-band companding effect.

dcshift *shift* [*limitergain*]
DC Shift the audio, with basic linear amplitude formula. This is most useful if your audio tends to not be centered around a value of 0. Shifting it back will allow you to get the most volume adjustments without clipping.

The first option is the *dcshift* value. It is a floating point number that indicates the amount to shift.

An optional *limitergain* can be specified as well. It should have a value much less than 1 (e.g. 0·05 or 0·02) and is used only on peaks to prevent clipping.

deemph

Apply a treble attenuation shelving filter to audio in audio-CD format. The frequency response of pre-emphasized recordings is rectified. The filter is defined in the standard document ISO 908.

This effect supports the **−−octave** global option.

See also the **bass** and **treble** shelving equalisation effects.

dither [*depth*]

Apply dithering to the audio. Dithering deliberately adds digital white noise to the signal in order to mask audible quantization effects that can occur if the output sample size is less than 24 bits. By default, the amount of noise added is ½ bit; the optional *depth* parameter is a (linear or voltage) multiplier of this amount.

This effect should not be followed by any other effect that affects the audio.

earwax

Makes audio easier to listen to on headphones. Adds 'cues' to audio in audio-CD format so that when listened to on headphones the stereo image is moved from inside your head (standard for headphones) to outside and in front of the listener (standard for speakers). See http://www.geocities.com/beinges for a full explanation.

echo *gain-in gain-out <delay decay>*

Add echoing to the audio. Each *delay decay* pair gives the delay in milliseconds and the decay (relative to gain-in) of that echo. Gain-out is the volume of the output.

echos *gain-in gain-out <delay decay>*

Add a sequence of echos to the audio. Each *delay decay* pair gives the delay in milliseconds and the decay (relative to gain-in) of that echo. Gain-out is the volume of the output.

equalizer *frequency width*[**q**|**o**|**h**] *gain*

Apply a two-pole peaking equalisation (EQ) filter. With this filter, the signal-level at and around a selected frequency can be increased or decreased, whilst (unlike band-pass and band-reject filters) that at all other frequencies is unchanged.

frequency gives the filter's central frequency in Hz, *width*, the band-width, as a Q-factor [2] (the default, or if appended with '**q**'), in octaves (if appended with '**o**'), or in Hz (if appended with '**h**'), and *gain* the required gain or attenuation in dB. Beware of **Clipping** when using a positive *gain*.

In order to produce complex equalisation curves, this effect can be given several times, each with a different central frequency.

The filter is described in detail in [1].

This effect supports the **−−octave** global option.

See also **bass** and **treble** for shelving equalisation effects.

fade [*type*] *fade-in-length* [*stop-time* [*fade-out-length*]]

Add a fade effect to the beginning, end, or both of the audio.

For fade-ins, this starts from the first sample and ramps the volume of the audio from 0 to full volume over *fade-in-length* seconds. Specify 0 seconds if no fade-in is wanted.

For fade-outs, the audio will be truncated at *stop-time* and the volume will be ramped from full volume down to 0 starting at *fade-out-length* seconds before the *stop-time*. If *fade-out-length* is not specified, it defaults to the same value as *fade-in-length*. No fade-out is performed if *stop-time* is not specified.

All times can be specified in either periods of time or sample counts. To specify time periods use the format hh:mm:ss.frac format. To specify using sample counts, specify the number of samples

and append the letter ‘s’ to the sample count (for example ‘8000s’).

An optional *type* can be specified to change the type of envelope. Choices are **q** for quarter of a sine wave, **h** for half a sine wave, **t** for linear slope, **l** for logarithmic, and **p** for inverted parabola. The default is a linear slope.

filter [*low*]−[*high*] [*window-len* [*beta*]]

Apply a sinc-windowed lowpass, highpass, or bandpass filter of given window length to the signal. *low* refers to the frequency of the lower 6dB corner of the filter. *high* refers to the frequency of the upper 6dB corner of the filter.

A low-pass filter is obtained by leaving *low* unspecified, or 0. A high-pass filter is obtained by leaving *high* unspecified, or 0, or greater than or equal to the Nyquist frequency.

The *window-len*, if unspecified, defaults to 128. Longer windows give a sharper cutoff, smaller windows a more gradual cutoff.

The *beta*, if unspecified, defaults to 16. This selects a Kaiser window. You can select a Nuttall window by specifying anything ≤ 2 here. For more discussion of beta, look under the **resample** effect.

flanger [*delay depth regen width speed shape phase interp*]

Apply a flanging effect to the audio. All parameters are optional (right to left).

	Range	**Default**	**Description**
delay	0 − 10	0	Base delay in milliseconds.
depth	0 − 10	2	Added swept delay in milliseconds.
regen	−95 − 95	0	Percentage regeneration (delayed signal feedback).
width	0 − 100	71	Percentage of delayed signal mixed with original.
speed	0·1 − 10	0·5	Sweeps per second (Hz).
shape		sin	Swept wave shape: **sine**|**triangle**.
phase	0 − 100	25	Swept wave percentage phase-shift for multi-channel (e.g. stereo) flange; 0 = 100 = same phase on each channel.
interp		lin	Digital delay-line interpolation: **linear**|**quadratic**.

See [3] for a detailed description of flanging.

highpass|**lowpass** [**-1**|**-2**] *frequency* [width[**q**|**o**|**h**]]

Apply a high-pass or low-pass filter with 3dB point *frequency*. The filter can be either single-pole (with **-1**), or double-pole (the default, or with **-2**). *width* applies only to double-pole filters and is the filter-width: as a Q-factor (the default, or if appended with ‘**q**’), in octaves (if appended with ‘**o**’), or in Hz (if appended with ‘**h**’); the default Q is 0·707 and gives a Butterworth response. The filters roll off at 6dB per pole per octave (20dB per pole per decade). The double-pole filters are described in detail in [1].

These effects support the **−−octave** global option.

See also **filter** for filters with a steeper roll-off.

lowpass [**-1**|**-2**] *frequency* [width[**q**|**o**|**h**]]

Apply a low-pass filter. See the description of the **highpass** effect for details.

mcompand "*attack1*,*decay1*{,*attack2*,*decay2*}
in-dB1,*out-dB1*{,*in-dB2*,*out-dB2*}
[*gain* [*initial-volume* [*delay*]]]" *xover-freq*

Multi-band compander is similar to the single band compander but the audio is first divided up into bands and then the compander is run on each band. See the **compand** effect for the definition of its options. Compand options are specified between double quotes and the crossover frequency for that band is specified separately with *xover-freq*. This can be repeated multiple times to create multiple bands.

mixer [**−l**|**−r**|**−f**|**−b**|**−1**|**−2**|**−3**|**−4**|*n*{,*n*}]

Reduce the number of audio channels by mixing or selecting channels, or increase the number of channels by duplicating channels. Note: this effect operates on the audio *channels* within the SoX effects processing chain; it should not be confused with the **−m** global option (where multiple *files* are mix-combined before entering the effects chain).

This effect is automatically used when the number of input channels differ from the number of output channels. When reducing the number of channels it is possible to manually specify the **mixer** effect and use the **−l**, **−r**, **−f**, **−b**, **−1**, **−2**, **−3**, **−4**, options to select only the left, right, front, back channel(s) or specific channel for the output instead of averaging the channels. The **−l**, and **−r** options will do averaging in quad-channel files so select the exact channel to prevent this.

The **mixer** effect can also be invoked with up to 16 numbers, separated by commas, which specify the proportion (0 = 0% and 1 = 100%) of each input channel that is to be mixed into each output channel. In two-channel mode, 4 numbers are given: l → l, l → r, r → l, and r → r, respectively. In four-channel mode, the first 4 numbers give the proportions for the left-front output channel, as follows: lf → lf, rf → lf, lb → lf, and rb → rf. The next 4 give the right-front output in the same order, then left-back and right-back.

It is also possible to use the 16 numbers to expand or reduce the channel count; just specify 0 for unused channels.

Finally, certain reduced combination of numbers can be specified for certain input/output channel combinations.

In Ch	Out Ch	Num	Mappings
2	1	2	l → l, r → l
2	2	1	adjust balance
4	1	4	lf → l, rf → l, lb → l, rb → l
4	2	2	lf → l&rf → r, lb → l&rb → r
4	4	1	adjust balance
4	4	2	front balance, back balance

noiseprof [*profile-file*]

Calculate a profile of the audio for use in noise reduction. See the description of the **noisered** effect for details.

noisered *profile-file* [*threshold*]

Noise reduction filter with profiling. This filter is moderately effective at removing consistent background noise such as hiss or hum. To use it, first run the **noiseprof** effect on a section of audio that ideally would contain silence but in fact contains noise. The **noiseprof** effect will write out a noise profile to *profile-file*, or to stdout if no *profile-file* is specified. If there is audio output on stdout then the profile will instead be directed to stderr.

To actually remove the noise, run SoX again with the *noisered* filter. The filter needs one parameter, *profile-file*, which contains the noise profile from **noiseprof**. *threshold* specifies how much noise should be removed, and may be between 0 and 1 with a default of 0·5. Higher values will remove more noise but present a greater likelihood of distorting the desired audio signal. Experiment with different threshold values to find the optimal one for your audio.

pad { *length*[**@***position*] }

Pad the audio with silence, at the beginning, the end, or any specified points through the audio. Both *length* and *position* can specify a time or, if appended with an ‘s’, a number of samples.

length is the amount of silence to insert and *position* the position in the input audio stream at which to insert it. Any number of lengths and positions may be specified, provided that a specified position is not less that the previous one. *position* is optional for the first and last lengths specified and if omitted correspond to the beginning and the end of the audio respectively. For example: **pad 1·5 1·5** adds 1·5 seconds of silence padding at each end of the audio, whilst **pad 4000s@3:00** inserts 4000 samples of silence 3 minutes into the audio. If silence is wanted only at the end of the audio, specify either the end position or specify a zero-length pad at the start.

pan *direction*

Pan the audio from one channel to another. This is done by changing the volume of the input channels so that it fades out on one channel and fades-in on another. If the number of input channels is different then the number of output channels then this effect tries to intelligently handle this. For instance, if the input contains 1 channel and the output contains 2 channels, then it will create the missing channel itself. The *direction* is a value from −1 to 1. −1 represents far left and 1 represents far right. Numbers in between will start the pan effect without totally muting the opposite channel.

phaser *gain-in gain-out delay decay speed* [**−s**|**−t**]

Add a phasing effect to the audio. delay/decay/speed gives the delay in milliseconds and the decay (relative to gain-in) with a modulation speed in Hz. The modulation is either sinusoidal (**−s**) or triangular (**−t**). The decay should be less than 0·5 to avoid feedback. Gain-out is the volume of the output.

pitch *shift* [*width interpolate fade*]

Change the pitch of file without affecting its duration by cross-fading shifted samples. *shift* is given in cents. Use a positive value to shift to treble, negative value to shift to bass. Default shift is 0. *width* of window is in ms. Default width is 20ms. Try 30ms to lower pitch, and 10ms to raise pitch. *interpolate* option, can be **cubic** or **linear**. Default is **cubic**. The *fade* option, can be **cos**, **hamming**, **linear** or **trapezoid**; the default is **cos**.

polyphase [**−w nut**|**ham**] [**−width** *n*] [**−cutoff** *c*]

Change the sampling rate using 'polyphase interpolation', a DSP algorithm. This method is relatively slow and memory intensive.

If the **−w** parameter is **nut**, then a Nuttall (˜90 dB stop-band) window will be used; **ham** selects a Hamming (˜43 dB stop-band) window. The default is Nuttall.

The **−width** parameter specifies the (approximate) width of the filter. The default is 1024 samples, which produces reasonable results.

The **−cutoff** value (*c*) specifies the filter cutoff frequency in terms of fraction of frequency bandwidth, also know as the Nyquist frequency. See the **resample** effect for further information on Nyquist frequency. If up-sampling, then this is the fraction of the original signal that should go through. If down-sampling, this is the fraction of the signal left after down-sampling. The default is 0·95.

See also **rabbit** and **resample** for other sample-rate changing effects.

rabbit [**−c0**|**−c1**|**−c2**|**−c3**|**−c4**]

Change the sampling rate using 'libsamplerate', also known as 'Secret Rabbit Code'. This effect is optional and must have been selected at compile time of SoX. See http://www.mega-nerd.com/SRC for details of the algorithms. Algorithms 0 through 2 are progressively faster and lower quality versions of the sinc algorithm; the default is **−c0**, which is probably the best quality algorithm for general use currently available in SoX. Algorithm 3 is zero-order hold, and 4 is linear interpolation.

See also **polyphase** and **resample** for other sample-rate changing effects, and see **resample** for more discussion of resampling.

repeat *count*

Repeat the entire audio *count* times. Requires disk space to store the data to be repeated. Note that repeating once yields two copies: the original audio and the repeated audio.

resample [**−qs**|**−q**|**−ql**] [*rolloff* [*beta*]]

Change the sampling rate using simulated analog filtration. Other rate changing effects available are **polyphase** and **rabbit**. There is a detailed analysis of **resample** and **polyphase** at http://leute.server.de/wilde/resample.html; see **rabbit** for a pointer to its own documentation.

By default, linear interpolation is used, with a window width about 45 samples at the lower of the two rates. This gives an accuracy of about 16 bits, but insufficient stop-band rejection in the case that you want to have roll-off greater than about 0·8 of the Nyquist frequency.

The **−q*** options will change the default values for roll-off and beta as well as use quadratic interpolation of filter coefficients, resulting in about 24 bits precision. The **−qs**, **−q**, or **−ql** options specify increased accuracy at the cost of lower execution speed. It is optional to specify roll-off and beta parameters when using the **−q*** options.

Following is a table of the reasonable defaults which are built-in to SoX:

Option	Window	Roll-off	Beta	Interpolation
(none)	45	0·80	16	linear
−qs	45	0·80	16	quadratic
−q	75	0·875	16	quadratic
−ql	149	0·94	16	quadratic

−qs, **−q**, or **−ql** use window lengths of 45, 75, or 149 samples, respectively, at the lower sample-rate of the two files. This means progressively sharper stop-band rejection, at proportionally slower execution times.

rolloff refers to the cut-off frequency of the low pass filter and is given in terms of the Nyquist frequency for the lower sample rate. rolloff therefore should be something between 0 and 1, in practise 0·8−0·95. The defaults are indicated above.

The *Nyquist frequency* is equal to half the sample rate. Logically, this is because the A/D converter needs at least 2 samples to detect 1 cycle at the Nyquist frequency. Frequencies higher then the Nyquist will actually appear as lower frequencies to the A/D converter and is called aliasing. Normally, A/D converts run the signal through a lowpass filter first to avoid these problems.

Similar problems will happen in software when reducing the sample rate of an audio file (frequencies above the new Nyquist frequency can be aliased to lower frequencies). Therefore, a good resample effect will remove all frequency information above the new Nyquist frequency.

The *rolloff* refers to how close to the Nyquist frequency this cutoff is, with closer being better. When increasing the sample rate of an audio file you would not expect to have any frequencies exist that are past the original Nyquist frequency. Because of resampling properties, it is common to have aliasing artifacts created above the old Nyquist frequency. In that case the *rolloff* refers to how close to the original Nyquist frequency to use a highpass filter to remove these artifacts, with closer also being better.

The *beta* parameter determines the type of filter window used. Any value greater than 2 is the beta for a Kaiser window. Beta ≤ 2 selects a Nuttall window. If unspecified, the default is a Kaiser window with beta 16.

In the case of Kaiser window (beta > 2), lower betas produce a somewhat faster transition from pass-band to stop-band, at the cost of noticeable artifacts. A beta of 16 is the default, beta less than 10 is not recommended. If you want a sharper cutoff, don't use low beta's, use a longer sample window. A Nuttall window is selected by specifying any 'beta' ≤ 2, and the Nuttall window has somewhat steeper cutoff than the default Kaiser window. You will probably not need to use the beta parameter at all, unless you are just curious about comparing the effects of Nuttall vs.

Kaiser windows.

This is the default effect if the two files have different sampling rates. Default parameters are, as indicated above, Kaiser window of length 45, roll-off 0·80, beta 16, linear interpolation.

Note: **−qs** is only slightly slower, but more accurate for 16-bit or higher precision.

Note: In many cases of up-sampling, no interpolation is needed, as exact filter coefficients can be computed in a reasonable amount of space. To be precise, this is done when

$$\text{input-rate} < \text{output-rate}$$
$$\text{and}$$
$$\text{output-rate} \div \gcd(\text{input-rate}, \text{output-rate}) \leq 511$$

reverb *gain-out reverb-time <delay>*

Add reverberation to the audio. Each *delay* is given in milliseconds and its feedback is depending on the *reverb-time* in milliseconds. Each *delay* should be in the range of half to quarter of *reverb-time* to get a realistic reverberation. *gain-out* is the volume of the output.

reverse Reverse the audio completely. Requires disk space to store the data to be reversed.

silence *above-periods* [*duration threshold*[**d**|**%**] [*below-periods duration threshold*[**d**|**%**]]

Removes silence from the beginning, middle, or end of the audio. Silence is anything below a specified threshold.

The *above-periods* value is used to indicate if audio should be trimmed at the beginning of the audio. A value of zero indicates no silence should be trimmed from the beginning. When specifying an non-zero *above-periods*, it trims audio up until it finds non-silence. Normally, when trimming silence from beginning of audio the *above-periods* will be 1 but it can be increased to higher values to trim all audio up to a specific count of non-silence periods. For example, if you had an audio file with two songs that each contained 2 seconds of silence before the song, you could specify an *above-period* of 2 to strip out both silence periods and the first song.

When *above-periods* is non-zero, you must also specify a *duration* and *threshold*. *Duration* indications the amount of time that non-silence must be detected before it stops trimming audio. By increasing the duration, burst of noise can be treated as silence and trimmed off.

Threshold is used to indicate what sample value you should treat as silence. For digital audio, a value of 0 may be fine but for audio recorded from analog, you may wish to increase the value to account for background noise.

When optionally trimming silence from the end of the audio, you specify a *below-periods* count. In this case, *below-period* means to remove all audio after silence is detected. Normally, this will be a value 1 of but it can be increased to skip over periods of silence that are wanted. For example, if you have a song with 2 seconds of silence in the middle and 2 second at the end, you could set below-period to a value of 2 to skip over the silence in the middle of the audio.

For *below-periods*, *duration* specifies a period of silence that must exist before audio is not copied any more. By specifying a higher duration, silence that is wanted can be left in the audio. For example, if you have a song with an expected 1 second of silence in the middle and 2 seconds of silence at the end, a duration of 2 seconds could be used to skip over the middle silence.

Unfortunately, you must know the length of the silence at the end of your audio file to trim off silence reliably. A work around is to use the **silence** effect in combination with the **reverse** effect. By first reversing the audio, you can use the *above-periods* to reliably trim all audio from what looks like the front of the file. Then reverse the file again to get back to normal.

To remove silence from the middle of a file, specify a *below-periods* that is negative. This value is then treated as a positive value and is also used to indicate the effect should restart processing as specified by the *above-periods*, making it suitable for removing periods of silence in the middle of the audio.

The *period* counts are in units of samples. *Duration* counts may be in the format of hh:mm:ss.frac, or the exact count of samples. *Threshold* numbers may be suffixed with **d** to indicate the value is in decibels, or **%** to indicate a percentage of maximum value of the sample value (**0%** specifies pure digital silence).

speed *factor*[**c**]
Adjust the audio speed (pitch and tempo together). *factor* is either the ratio of the new speed to the old speed: greater than 1 speeds up, less than 1 slows down, or, if appended with the letter 'c', the number of cents (i.e. 100ths of a semitone) by which the pitch (and tempo) should be adjusted: greater than 0 increases, less than 0 decreases.

By default, the speed change is performed by the **resample** effect with its default parameters. For higher quality resampling, in addition to the **speed** effect, specify either the **resample** or the **rabbit** effect with appropriate parameters.

stat [**−s** *n*] [**−rms**] [**−freq**] [**−v**] [**−d**]
Do a statistical check on the input file, and print results on the standard error file. Audio is passed unmodified through the SoX processing chain.

The 'Volume Adjustment:' field in the statistics gives you the parameter to the **−v** *number* which will make the audio as loud as possible without clipping. Note: See the discussion on **Clipping** above for reasons why it is rarely a good idea to actually do this.

The option **−v** will print out the 'Volume Adjustment:' field's value only and return. This could be of use in scripts to auto convert the volume.

The **−s** option is used to scale the input data by a given factor. The default value of *n* is the max value of a signed long variable (0x7fffffff). Internal effects always work with signed long PCM data and so the value should relate to this fact.

The **−rms** option will convert all output average values to 'root mean square' format.

The **−freq** option calculates the input's power spectrum and prints it to standard error.

There is also an optional parameter **−d** that will print out a hex dump of the audio from the internal buffer that is in 32-bit signed PCM data. This is mainly only of use in tracking down endian problems that creep in to SoX on cross-platform versions.

stretch *factor* [*window fade shift fading*]
Time stretch the audio by the given factor. Changes duration without affecting the pitch. *factor* of stretching: >1 lengthen, <1 shorten duration. *window* size is in ms. Default is 20ms. The *fade* option, can be 'lin'. *shift* ratio, in [0 1]. Default depends on stretch factor. 1 to shorten, 0·8 to lengthen. The *fading* ratio, in [0 0·5]. The amount of a fade's default depends on *factor* and *shift*.

swap [*1 2 | 1 2 3 4*]
Swap channels in multi-channel audio files. Optionally, you may specify the channel order you would like the output in. This defaults to output channel 2 and then 1 for stereo and 2, 1, 4, 3 for quad-channels. An interesting feature is that you may duplicate a given channel by overwriting another. This is done by repeating an output channel on the command-line. For example, **swap 2 2** will overwrite channel 1 with channel 2; creating a stereo file with both channels containing the same audio.

synth [*len*] {[*type*] [*combine*] [*freq*[*−freq2*]] [*off*] [*ph*] [*p1*] [*p2*] [*p3*]}
This effect can be used to generate fixed or swept frequency audio tones with various wave shapes, or to generate wide-band noise of various 'colours'. Multiple synth effects can be cascaded to produce more complex waveforms; at each stage it is possible to choose whether the generated waveform will be mixed with, or modulated onto the output from the previous stage. Audio for each channel in a multi-channel audio file can be synthesised independently.

Though this effect is used to generate audio, an input file must still be given, the characteristics of

which will be used to set the synthesised audio length, the number of channels, and the sampling rate; however, since the input file's audio is not normally needed, a 'null file' (with the special name **-n**) is often given instead (and the length specified as a parameter to **synth** or by another given effect that can has an associated length).

For example, the following produces a 3 second, 44·1 kHz, stereo audio file containing a sine-wave swept from 300 to 3300 Hz:

sox −n output.au synth 3 sine 300−3300

and this produces an 8 kHz mono version:

sox −r 8000 −c 1 −n output.au synth 3 sine 300−3300

Multiple channels can be synthesised by specifying the set of parameters shown between braces multiple times; the following puts the swept tone in the left channel and adds 'brown' noise in the right:

sox −n output.au synth 3 sine 300−3300 brownnoise

The following example shows how two synth effects can be cascaded to create a more complex waveform:

sox −n output.au synth 0·5 sine 200−500 synth 0·5 sine fmod 700−100

Frequencies can also be given as a number of musical semitones relative to 'middle A' (440 Hz) by prefixing a '%' character; for example, the following could be used to help tune a guitar's 'E' strings:

play −n synth sine %−17

N.B. This effect generates audio at maximum volume, which means that there is a high chance of clipping when using the audio subsequently, so in most cases, you will want to follow this effect with the **vol** effect to prevent this from happening. (See also **Clipping** above.)

A detailed description of each **synth** parameter follows:

len is the length of audio to synthesise expressed as a time or as a number of samples; 0=input-length, default=0.

The format for specifying lengths in time is hh:mm:ss.frac. The format for specifying sample counts is the number of samples with the letter 's' appended to it.

type is one of sine, square, triangle, sawtooth, trapezium, exp, [white]noise, pinknoise, brown-noise; default=sine

combine is one of create, mix, amod (amplitude modulation), fmod (frequency modulation); default=create

freq/*freq2* are the frequencies at the beginning/end of synthesis in Hz or, if preceded with '%', semitones relative to A (440 Hz); for both, default=%0. If *freq2* is given, then *len* must also have been given. Not used for noise.

off is the bias (DC-offset) of the signal in percent; default=0.

ph is the phase shift in percentage of 1 cycle; default=0. Not used for noise.

p1 is the percentage of each cycle that is 'on' (square), or 'rising' (triangle, exp, trapezium); default=50 (square, triangle, exp), default=10 (trapezium).

p2 (trapezium): the percentage through each cycle at which 'falling' begins; default=50. exp: the amplitude in percent; default=100.

p3 (trapezium): the percentage through each cycle at which 'falling' ends; default=60.

treble *gain* [*frequency* [*width*[**s**|**h**|**o**|**q**]]]
Apply a treble tone-control effect. See the description of the **bass** effect for details.

tremolo *speed* [*depth*]
Apply a tremolo (low frequency amplitude modulation) effect to the audio. The tremolo frequency in Hz is given by *speed*, and the depth as a percentage by *depth* (default 40).

Note: This effect is a special case of the **synth** effect.

trim *start* [*length*]
Trim can trim off unwanted audio from the beginning and end of the audio. Audio is not sent to the output stream until the *start* location is reached.

The optional *length* parameter tells the number of samples to output after the *start* sample and is used to trim off the back side of the audio. Using a value of 0 for the *start* parameter will allow trimming off the back side only.

Both options can be specified using either an amount of time or an exact count of samples. The format for specifying lengths in time is hh:mm:ss.frac. A start value of 1:30·5 will not start until 1 minute, thirty and ½ seconds into the audio. The format for specifying sample counts is the number of samples with the letter 's' appended to it. A value of 8000s will wait until 8000 samples are read before starting to process audio.

vol *gain* [*type* [*limitergain*]]
Apply an amplification or an attenuation to the audio signal. Unlike the **−v** option (which is used for balancing multiple input files as they enter the SoX effects processing chain), **vol** is an effect like any other so can be applied anywhere, and several times if necessary, during the processing chain.

The amount to change the volume is given by *gain* which is interpreted, according to the given *type*, as follows: if *type* is **amplitude** (or is omitted), then *gain* is an amplitude (i.e. voltage or linear) ratio, if **power**, then a power (i.e. wattage or voltage-squared) ratio, and if **dB**, then a power change in dB.

When *type* is **amplitude** or **power**, a *gain* of 1 leaves the volume unchanged, less than 1 decreases it, and greater than 1 increases it; a negative *gain* inverts the audio signal in addition to adjusting its volume.

When *type* is **dB**, a *gain* of 0 leaves the volume unchanged, less than 0 decreases it, and greater than 0 increases it.

See [4] for a detailed discussion on electrical (and hence audio signal) voltage and power ratios.

Beware of **Clipping** when the increasing the volume.

An optional *limitergain* value can be specified and should be a value much less than 1 (e.g. 0·05 or 0·02) and is used only on peaks to prevent clipping. Not specifying this parameter will cause no limiter to be used. In verbose mode, this effect will display the percentage of the audio that needed to be limited.

Deprecated Effects

The following effects have been renamed or have their functionality included in another effect. They continue to work in this version of SoX but may be removed in future.

avg [**−l**|**−r**|**−f**|**−b**|**−1**|**−2**|**−3**|**−4**|*n*{,*n*}]
Reduce the number of audio channels by mixing or selecting channels, or duplicate channels to increase the number of channels. This effect is just an alias of the **mixer** effect and is retained for backwards compatibility only.

highp *frequency*
Apply a high-pass filter. This effect is just an alias for the **highpass** effect used with its **-1** option; it is retained for backwards compatibility only.

lowp *frequency*
Apply a low-pass filter. This effect is just an alias for the **lowpass** effect used with its **-1** option; it is retained for backwards compatibility only.

mask [*depth*]
This effect is just a deprecated alias for the **dither** effect, left for historical reasons.

pick [**−l**|**−r**|**−f**|**−b**|**−1**|**−2**|**−3**|**−4**|*n*{**,***n*}]
Pick a subset of channels to be copied into the output file. This effect is just an alias of the **mixer** effect and is retained for backwards compatibility only.

rate Does the same as **resample** with no parameters; it exists for backwards compatibility.

vibro *speed* [*depth*]
This is a deprecated alias for the **tremolo** effect. It differs in that the depth parameter ranges from 0 to 1 and defaults to 0·5.

DIAGNOSTICS

Exit status is 0 for no error, 1 if there is a problem with the command-line parameters, or 2 if an error occurs during file processing.

BUGS

Please report any bugs found in this version of SoX to the mailing list (sox-users@lists.sourceforge.net).

SEE ALSO

soxexam(7), **libst**(3)

The SoX web page at http://sox.sourceforge.net

References

[1] R. Bristow-Johnson, *Cookbook formulae for audio EQ biquad filter coefficients*, http://musicdsp.org/files/Audio-EQ-Cookbook.txt

[2] Wikipedia, *Q-factor*, http://en.wikipedia.org/wiki/Q_factor

[3] Scott Lehman, *Flanging*, http://harmony-central.com/Effects/Articles/Flanging

[4] Wikipedia, *Decibel*, http://en.wikipedia.org/wiki/Decibel

LICENSE

AUTHORS

Chris Bagwell (cbagwell@users.sourceforge.net). Other authors and contributors are listed in the AUTHORS file that is distributed with the source code.