sox

Benjamin Niemann

COLLABORATORS			
	TITLE : sox		
ACTION	*NAME*	*DATE*	*SIGNATURE*
WRITTEN BY	Benjamin Niemann	January 19, 2025	

REVISION HISTORY			
NUMBER	DATE	DESCRIPTION	NAME

Contents

Chapter 1

SOX

1.1 Welcome

```
This is SOX 12.16 as I found it at http://home.sprynet.com/~cbagwell/sox.html.
I compiled it for m68000 and m68020 using gcc 2.95 without modifications
to the base source.

It requires ixemul.library V48.

install
readme
manual
tips
cheats
effects cheats
```

1.2 README

```
                SoX: Sound eXchange


SoX (also known as Sound eXchange) translates sound samples between different
file formats, and optionally performs various sound effects.

This release understands:

  o Raw files in various binary formats
  o Raw textual data
  o Microsoft .WAV files
    o PCM, u-law, a-law
    o MS ADPCM (Read only)
    o IMA ADPCM (Read only)
  o MAUD files
  o Sound Blaster .VOC files
  o IRCAM SoundFile files
  o SUN .au files
    o PCM, u-law, a-law
    o G7xx ADPCM files (read only)
```

- mutant DEC .au files
- Apple/SGI AIFF files
- CD-R (music CD format)
- Macintosh HCOM files
- Sounder files
- NeXT .snd files
- Soundtool (DOS) files
- Psion (palmtop) A-law files

The sound effects include:

- Channel Averaging
- Band-pass filter
- Chorus effect
- Cut out loop samples
- Add an echo
- Add a sequence of echos
- Apply a flanger effect
- Apply a high-pass filter
- Apply a low-pass filter
- Display a list of loops in a file
- Add masking noise to a signal
- Apply a phaser effect
- Convert from stereo to mono
- Change sampling rates using several different algorithms.
- Apply a reverb effect
- Reverse the sound samples (to search for Satanic messages ;-)
- Convert from mono to stereo
- Swap stereo channels
- Display general stats on a sound sample
- Add the world-famous Fender Vibro-Champ effect

Big news! Lots of new effects have been added. This includes most the popular "Guitar Effects" talked about in the same named FAQ available.

The 'resample' and 'polyphase' effect does high-grade signal rate changes using real signal theory. Yes, it's very slow.

History:

This is the 12th release, Patchlevel 16 of the Sound Tools. SoX was originally written and maintained by Lance Norskog but unfortunetly he has stopped maintaining it since 1995. I, Chris Bagwell (cbagwell@sprynet.com), have started maintaining it since 1996 to the present.

Caveats:

SoX is intended as the Swiss Army knife of sound processing tools. It doesn't do anything very well, but sooner or later it comes in very handy. SoX is really only usable day-to-day if you hide the wacky options with one-line shell scripts.

Installing:

Unless your using a precompiled binary version, you will need to compile SoX as described in the INSTALL file. Please read that file for further

instructions.

Now, read TIPS, CHEAT.eft and CHEAT. These give a background on how
SoX deals with sound files and how to convert this format
to that format, and apply various effects.

SoX uses file suffices to determine the nature of a sound sample file.
If it finds the suffix in its list, it uses the appropriate read
or write handler to deal with that file. You may override the suffix
by giving a different type via the '-t type' argument. See the manual
page for more information.

SoX has an auto-detect feature that attempts to figure out
the nature of an unmarked sound sample. It works very well.
This feature is used if you specify '-t auto' for the file type.

I hope to inspire the creation of a common base of sound processing
tools for computer multimedia work, similar to the PBM toolkit for
image manipulation.

Sound Tools may be used for any purpose. Source
distributions must include the copyright notices. Binary
distributions must include acknowledgements to the creators.
Files are copyright by their respective authors.

If you have bug fixes/enhancements, please send it to me as I would like
to coordinate the releases. Please document your changes. I don't
possess every kind of computer currently sold, and SoX is now beyond
the phase where I can understand and test most of your contributions.

The majority of SoX features and source code are contributed
by you the user. Thank you very much for making SoX a success!

Creator:
Lance Norskog thinman@meer.net (inactive currently)

Mantainer:
Chris Bagwell cbagwell@sprynet.com

Contributors:
Juergen Mueller jmueller@uia.ua.ac.be
chorus, echo, echos, flanger, phaser, and reverb
effects.
Guido Van Rossum guido@cwi.nl
AU, AIFF, AUTO, HCOM, reverse,
many bug fixes
Jef Poskanzer jef@well.sf.ca.us
original code for u-law and delay line
Bill Neisius bill%solaria@hac2arpa.hac.com
DOS port, 8SVX, Sounder, Soundtool formats
Apollo fixes, stat with auto-picker
Rick Richardson rick@digibd.com
WAV and SB driver handlers, fixes
David Champion dgc3@midway.uchicago.edu
Amiga port
Pace Willisson pace@blitz.com
Fixes for ESIX

```
Leigh Smith             leigh@psychokiller.dialix.oz.au
        SMP and comment movement support.
        AIFF Loop/MIDI support
David Sanderson         dws@ssec.wisc.edu
        AIX3.1 fixes
Glenn Lewis             glewis@pcocd2.intel.com
        AIFF chunking fixes
Brian Campbell          brianc@quantum.qnx.com
        QNX port and 16-bit fixes
Chris Adams             gt8741@prism.gatech.edu
        DOS port fixes
John Kohl               jtkohl@kolvir.elcr.ca.us
        BSD386 port, VOC stereo support
Ken Kubo                ken@hmcvax.claremont.edu
        VMS port, VOC stereo support
Frank Gadegast          <phade@cs.tu-berlin.de>
        Microsoft C 7.0 & C Borland 3.0 ports
David Elliot            <dce@scmc.sony.com>
        CD-R format support
David Sears             <dns@essnj3.essnjay.com>
        Linux support
Tom Littlejohn          <tlit@seq1.loc.gov>
        Raw textual data
Boisy G. Pitre          boisy@microware.com
        OS9 port
Sun Microsystems, Guido Van Rossum
        CCITT G.711, G.721, G.723 implementation
Graeme Gill             graeme@labtam.labtam.oz.au
        A-LAW format, Good .WAV handling,
        avg channel expansion
Allen Grider            grider@hfsi.hfsi.com
        VOC stereo mode, WAV file handling
Michel Fingerhut        Michel.Fingerhut@ircam.fr
        Upgrade 'sf' format to current IRCAM format.
        Float file support.
Chris Knight
        Achimedes Acorn support
Richard Caley           R.Caley@ed.ac.uk
        Psion WVE handler
Lutz Vieweg             lkv@mania.RoBIN.de
        MAUD (Amiga) file handler
Tim Gardner             timg@tpi.com
        Windows NT port for V7
Jimen Ching             jiching@wiliki.eng.hawaii.edu
        Libst porting bugs
Lauren Weinstein        lauren@vortex.com
        DOS porting, scripts, professional use
Chris Bagwell           cbagwell@sprynet.com
        OSS and Sun players, bugfixes, ADPCM support,
        patch collection and maintance.
(your name could be here, too)
(I've probably lost a few, and several people fixed
 the same bugs.)
```

1.3 TIPS

```
SOX usage:
        sox [options] from-file-args to-file-args [ effect [effect-args]]

First off: the -V option makes SOX print out its idea of
what it is doing.  -V is your friend.

        sox -V from-file-args to-file-args

From-file-args and to-file-args are the same.
They are a series of options followed by a file name.
The suffix on the file name usually is the file format type.
The '-t xx' option overrides this and tells sox
the the file format is 'xx'.  The '-u/-s/-U' arguments
say that the file is in unsigned, signed, or u-law format.
The '-b/-w' arguments say that the file is in byte- or
word-size (2 byte) samples.  The '-r number' argument
says that the sample rate of the file is 'number'.

The extensions ub, uw, sb, sw, and ul correspond
to raw data files of formats unsigned byte, unsigned
word, signed byte, signed word, and u-law byte.
Thus, '-t ul' is shorthand for '-t raw -U -b'.

These conversions clip data and thus reduce sound quality,
so be careful:

        Word to u-law.
        Word to byte.
        U-law to byte.
        Reduction in sample rate.

Any reduction in the sample data rate loses information
and adds noise.  An increase in the data rate doesn't
lose much information, but does add noise.  See the
note below on low-pass filtering.

To convert U-law to something else without clipping,
you'll have to convert it to (signed or unsigned) words,
which will double the size of the file.

AUTO files:
The 'AUTO' file type reads an unknown file and
attempts to discern its binary format.

AIFF files:
AIFF files come with complete headers and other
info.  They can in fact have multiple sound
chunks and picture chunks.  SOX only reads
the first sound chunk.

WAV files:
WAVs use the RIFF format, which is Microsoft's
needless imitation of AIFF.  See above comments.
```

```
AIFF and RIFF files need their own librarian
programs; SOX can only do a small fraction of
what they need.

It's best if you can copy or store files in
AIFF or WAV format.  The sample rate and
binary format are marked; also comments may
be added to the file.

SUN AU files:
Most AU files you find are in 8khz 8-bit u-law format.
This format was the first sound hardware SUN made available.
Some of the files have correct headers; some do not.
If the file has the header, this should convert it to
another format:

        sox file.au to-file-args

If not, this reads a raw u-law 8khz file:

        sox -t ul -r 8000 file.au to-file-args

To convert a file to an old-style SUN .au file:

        sox from-file-args -r 8000 -U -b file.au

AU format can have any speed and several data sizes;
you need to specify '-r 8000 -U -b' to force SOX to
use the old SUN format.

Mac files:
Mac files come in .snd, .aiff, and .hcom formats,
among others; these are the most common.

SND files are in unsigned byte format with no
header.  They are either 11025, 22050, or 44100 hz.
The speed seems to be a "resource" and doesn't
get transported to Unix when the files are.
Thus, you just have to know.

        sox -r 11025 -t ub file.snd to-file-args
        sox from-file-args -r 11025 -t ub file.snd

PC files:
There are several PC sound file formats. VOC is
common; it has headers.  SND and SNDR are for
some DOS sound package; I don't know much about them.
WAV is the official Microsoft Windows format.
WAV has format options for compressed sound;
SOX doesn't implement this yet.


Effects:
A sound effect may be applied to the sound sample
while it is being copied from one file to another.
Copy is the default effect; i.e. do nothing.
Changing the sample rate requires the 'rate'
```

```
effect.  This applies a simple linear interpolation
to the sample.  This is a poor-quality sample
changer.  After doing a rate conversion,
you should try doing a low-pass filter to throw
away some of the induced noise.  Pick a 'center'
frequency about 85% of the lower of the two
frequencies, or 42.5% of the lower of the
two sample rates.  (The maximum frequency
in a sample is 1/2 of the sample rate).

        sox -r 8000 file.xx -r 22050 tmp.yy
        sox tmp.yy file.yy lowp 3400
or:
        sox -r 44100 file.xx -r 22050 tmp.yy
        sox tmp.yy file.yy lowp 9592

Listen to both tmp.yy and file.yy and see if
the low-pass filter helps.  Be sure to do the
low-pass filter before clipping the data to
a smaller binary word size.  Say you have a 16-bit
CD-quality (44100 hz) AIFF file that you want
to convert to a Mac sound resource:

        sox -r 44100 file.aiff -r 11025 tmp.sw
        sox tmp.sw -t ub file.mac lowp 9371

not:

        sox -r 44100 file.aiff -r 11025 tmp.ub
        sox tmp.ub -t ub file.mac lowp 9371

because you want to do the low-pass filter while
you still have sixteen-bit data.
```

1.4 INSTALL

```
INSTALLATIONS
-------------

1) rename sox-m68000 or sox-m68020 to sox depending
   on your CPU (use sox-m68000, if you don't know which).

2) copy sox to C:
   or add PATH dir to your user-startup, where
   dir is the directory of sox.
```

1.5 CHEATS

```
CHEAT SHEET
-----------

This is a cheat sheet of examples using SOX to do various common
```

sound file conversions. The file format examples are starting
to become dated. Any offers to update this document to explain
the ends and outs of each format would be appreciated.

In general, sox will attempt to take an input sound file format and
convert it to a new file format using a similar data types and sample
rates. For instance "sox monkey.au monkey.wav" would try and convert
the mono 8000Hz u-law .au file to a 8000Hz u-law .wav file.

If an output format doesn't support the same data types as the input
file then Sox will generally select a default format to save it in.
You can select a data type of your choice using command line options.
You can also override data type values to have a output file of
higher or lower percision data (and thus higher or lower file size).

Most file formats that contain complete headers will automatically
convert to a similar format. This means .wav, .aiff, and .voc files
will readily convert to each other without the need of complex
command lines.

If you create a sound file and you can not play it, check to make
sure your sound card to play a file using this data type.

SOX is great to use along with other command line programs. The
currently most used example is to use mpg123 to convert mp3 files
in to wav files. The following command line will do this:

```
mpg123 -b 10000 -s filename.mp3 | sox -t raw -r 44100 -s -w -c 2 - filename.wav
```

The SUN examples below all assume you have the old SUN voice-quality 8khz
u-law hardware. If you do then you will want to have all your .au
files in this format so that you cat do thinks like
"cat soundfile.au > /dev/audio" and you will hear a good file.
If the .AU file doesn't have a proper header, then you'll need the second
command line to let sox know the values. If the .AU has a proper
header then you can remove the "-r 8000 -U -b" in front of
"file.au".

SUN .au to Mac .snd:

```
        sox file.au -r 11025 -t ub file.snd
```
or:
```
        sox -t ul -r 8000 file.au -r 11025 -t ub file.snd
```

When you copy the file to the Mac, you'll have to set
the sample rate by hand.

Mac .snd to SUN .au

```
        sox -r 11025 -t ub file.snd -r 8000 -U -b file.au
```

The Mac file might also be at sample rates 5012, 22050, or 44100.

PC .voc to SUN .au

```
        sox file.voc -r 8000 -U -b file.au
```

```
SUN .au to PC .voc

        sox file.au file.voc
or:
        sox -r 8000 -t ul file.au file.voc

SUN .au to WAV - without clipping

        sox file.au -s -w file.wav
or:
        sox -t ul -r 8000 file.au -s -w file.wav

WAV to SUN .au

        sox file.wav -r 8000 -U -b file.au

WAV to VOC
        sox file.wav -u -b file.voc

VOC to WAV
        sox file.voc file.wav

Any file to SUN .au

sox -t auto file.X -c 1 -t aiff - |  sox -t aiff - -r 8000 -U -b -t au file.au

Just convert file format without making a disk file.
Example: convert input stream in AIFF format to output stream in WAV format:

sox -t aiff - -t wav -

Some people try to put this kind of command in scripts.

It is important to understand how the internals of Sox work when
working with compressed audio, including u-law, a-law, ADPCM, or GSM.
Sox takes ALL input data types and converts them to uncompressed
32-bit signed data.  It will then convert this internal version into
the requested output format.  This means unneeded noise can be introduced
from decompressing data and then recompressing, such as would happen
when reading u-law data and writing back out u-law data.  If possible,
specify the output data to be uncompressed PCM.

Under DOS, you can convert several using something similar to the
following command line:

FOR %X IN (*.RAW) DO sox -r 11025 -w -s -t raw $X $X.wav

Good luck!
```

1.6 FX CHEATS

```
This is a cheat sheet of examples using SOX to
add effects on sound files.

Introduction:
```

The core problem is that you need some experience in using effects
in order to say "that any old sound file sounds with effects
absolutely hip". There isn't any rule-based system which tell you
the correct setting of all the parameters for every effect.
But after some time you will become an expert in using effects.

Here are some examples which can be used with any music sample.
(For a sample where only a single instrument is playing, extreme
parameter setting may make well-known "typically" or "classical"
sounds. Likewise, for drums, vocals or guitars.)
Single effects will be explained and some given parameter settings
that can be used to understand the theorie by listening to the sound file
with the added effect.

Using multiple effects in parallel or in sequel can result either
in very perfect sound or (mostly) in a dramatic overloading in
variations of sounds such that your ear may follow the sound but
you will feel unsatisfied. Hence, for the first time using effects
try to compose them as less as possible. We don't regard the
composition of effects in the examples because to many combinations
are possible and you really need a very fast maschine and a lot of
memory to play them in real-time.

And real-time playing of sounds will speed up learning the parameter
setting.

Basically, we will use the "play" front-end of SOX since it is easier
to listen sounds coming out of the speaker or earphone instead
of looking at cryptical data in sound files.

For easy listening of file.xxx ("xxx" is any sound format):

```
play file.xxx effect-name effect-parameters
```

Or more SOX-like (for "dsp" output):

```
sox file.xxx -t ossdsp -w -s /dev/dsp effect-name effect-parameters
```

or (for "au" output):

```
sox file.xxx -t sunau -w -s /dev/audio effect-name effect-parameters
```

And for date freaks:

```
sox file.xxx file.yyy effect-name effect-parameters
```

Additional options can be used. However, in this case, for real-time
playing you'll need a very fast machine.

Notes:

I played all examples in real-time on a Pentium 100 with 32 Mb and
Linux 2.0.30 using a self-recorded sample (3:15 min long in "wav"
format with 44.1 kHz sample rate and stereo 16 bit).
The sample should not contain any of the effects. However,
if you take any recording of a sound track from radio or tape or cd,

and it sounds like a live concert or ten people are playing the same
rhythm with their drums or funky-groves, then take any other sample.
(Typically, less then four different intruments and no synthesizer
in the sample is suitable. Likewise, the combination vocal, drums, bass
and guitar.)

Effects:

Echo

An echo effect can be naturally found in the mountains, standing somewhere
on a moutain and shouting a single word will result in one or more repetitions
of the word (if not, turn a bit around ant try next, or climb to the next
mountain).

However, the time difference between shouting and repeating is the delay
(time), its loudness is the decay. Multiple echos can have different delays and
decays.

Very popular is using echos to play an instrument with itself together, like
some guitar players (Brain May from Queen) or vocalists are doing.
For music samples of more than one instrument, echo can be used to add a
second sample shortly after the original one.
This will sound as doubling the number of instruments playing the same sample:

```
play file.xxx echo 0.8 0.88 60.0 0.4
```

If the delay is very short then it sound like a (metallic) roboter playing
music:

```
play file.xxx echo 0.8 0.88 6.0 0.4
```

Longer delay will sound like a open air concert in the mountains:

```
play file.xxx echo 0.8 0.9 1000.0 0.3
```

One mountain more, and:

```
play file.xxx echo 0.8 0.9 1000.0 0.3 1800.0 0.25
```

Echos

Like the echo effect, echos stand for "ECHO in Sequel", that is the first echos
takes the input, the second the input and the first echos, the third the input
and the first and the second echos, ... and so on.
Care should be taken using many echos (see introduction); a single echos
has the same effect as a single echo.
The sample will be bounced twice in symmetric echos:

```
play file.xxx echos 0.8 0.7 700.0 0.25 700.0 0.3
```

The sample will be bounced twice in asymmetric echos:

```
play file.xxx echos 0.8 0.7 700.0 0.25 900.0 0.3
```

The sample will sound as played in a garage:

```
play file.xxx echos 0.8 0.7 40.0 0.25 63.0 0.3
```

Chorus

The chorus effect has its name because it will often be used to make a single vocal sound like a chorus. But it can be applied to other instrument samples too.

It works like the echo effect with a short delay, but the delay isn't constant. The delay is varied using a sinodial or triangular modulation. The modulation depth defines the range the modulated delay is played before or after the delay. Hence the delayed sound will sound slower or faster, that is the delayed sound tuned around the original one, like in a chorus where some vocal are a bit out of tune.

The typical delay is around 40ms to 60ms, the speed of the modualtion is best near 0.25Hz and the modulation depth around 2ms.

A single delay will make the sample more overloaded:

```
play file.xxx chorus 0.7 0.9 55.0 0.4 0.25 2.0 -t
```

Two delays of the original samples sound like this:

```
play file.xxx chorus 0.6 0.9 50.0 0.4 0.25 2.0 -t 60.0 0.32 0.4 1.3 -s
```

A big chorus of the sample is (three additional samples):

```
play file.xxx chorus 0.5 0.9 50.0 0.4 0.25 2.0 -t 60.0 0.32 0.4 2.3 -t \
        40.0 0.3 0.3 1.3 -s
```

Flanger

The flanger effect is like the chorus effect, but the delay varies between 0ms and maximal 5ms. It sound like wind blowing, sometimes faster or slower including changes of the speed.

The flanger effect is widely used in funk and soul music, where the guitar sound varies frequently slow or a bit faster.

The typical delay is around 3ms to 5ms, the speed of the modulation is best near 0.5Hz.

Now, let's groove the sample:

```
play file.xxx flanger 0.6 0.87 3.0 0.9 0.5 -s
```

listen carefully between the difference of sinodial and triangular modulation:

```
play file.xxx flanger 0.6 0.87 3.0 0.9 0.5 -t
```

If the decay is a bit lower, than the effect sounds more popular:

```
play file.xxx flanger 0.8 0.88 3.0 0.4 0.5 -t
```

The drunken loundspeaker system:

```
play file.xxx flanger 0.9 0.9 4.0 0.23 1.3 -s
```

Reverb

The reverb effect is often used in audience hall which are to small or to many visitors disturb the reflection of sound at the walls to make the sound played more monumental. You can try the reverb effect in your bathroom or garage or sport halls by shouting loud some words. You'll hear the words reflected from the walls.

The biggest problem in using the reverb effect is the correct setting of the (wall) delays such that the sound is relistic an doesn't sound like music playing in a tin or overloaded feedback distroys any illusion of any big hall. To help you for much realisitc reverb effects, you should decide first, how long the reverb should take place until it is not loud enough to be registered by your ears. This is be done by the reverb time "t", in small halls 200ms in bigger one 1000ms, if you like. Clearly, the walls of such a hall aren't far away, so you should define its setting be given every wall its delay time. However, if the wall is to far eway for the reverb time, you won't hear the reverb, so the nearest wall will be best "t/4" delay and the farest "t/2". You can try other distances as well, but it won't sound very realistic. The walls shouldn't stand to close to each other and not in a multiple interger distance to each other (so avoid wall like: 200.0 and 202.0, or something like 100.0 and 200.0).

Since audience halls do have a lot of walls, we will start designing one beginning with one wall:

```
play file.xxx reverb 1.0 600.0 180.0
```

One wall more:

```
play file.xxx reverb 1.0 600.0 180.0 200.0
```

Next two walls:

```
play file.xxx reverb 1.0 600.0 180.0 200.0 220.0 240.0
```

Now, why not a futuristic hall with six walls:

```
play file.xxx reverb 1.0 600.0 180.0 200.0 220.0 240.0 280.0 300.0
```

If you run out of machine power or memory, then stop as much applications as possible (every interupt will consume a lot of cpu time which for bigger halls is absolutely neccessary).

Phaser

The phaser effect is like the flanger effect, but it uses a reverb instead of an echo and does phase shifting. You'll hear the difference in the examples comparing both effects (simply change the effect name).

```
The delay modulation can be done sinodial or triangular, preferable is the
later one for multiple instruments playing. For single instrument sounds
the sinodial phaser effect will give a sharper phasing effect.
The decay shouln't be to close to 1.0 which will cause dramatic feedback.
A good range is about 0.5 to 0.1 for the decay.

We will take a parameter setting as for the flanger before ( gain-out is
lower since feedback can raise the output dramatically ):

        play file.xxx phaser 0.8 0.74 3.0 0.4 0.5 -t

The drunken loundspeaker system ( now less alkohol ):

        play file.xxx phaser 0.9 0.85 4.0 0.23 1.3 -s

A popular sound of the sample is as follows:

        play file.xxx phaser 0.89 0.85 1.0 0.24 2.0 -t

The sample sounds if ten springs are in your ears:

        play file.xxx phaser 0.6 0.66 3.0 0.6 2.0 -t

Other effects ( copy, rate, avg, stat, vibro, lowp, highp, band, reverb )
-------------

The other effects are simply to use. However, an "easy to use manual" should
be given here.

More effects ( to do ! )
------------

There are a lot of effects around like noise gates, compressors, waw-waw,
stereo effects and so on. They should be implemented making SOX to be more
useful in sound mixing technics coming together with a great varity of
different sound effects.

Combining effects be using then in parallel or sequel on different channels
needs some easy mechanism which is real-time stable.

Really missing, is the changing of the parameters, starting and stoping of
effects while playing samples in real-time!


Good luck and have fun with all the effects!

        Juergen Mueller         (jmueller@uia.ua.ac.be)
```

1.7 MANUAL

```
NAME
       sox - Sound eXchange : universal sound sample translator

SYNOPSIS
```

```
sox infile outfile
sox infile outfile [ effect [ effect options ... ] ]
sox infile -e effect [ effect options ... ]
sox  [ general options  ] [ format options  ] ifile [ for-
mat options  ] ofile [ effect [ effect options ... ] ]

General options: [ -e ] [ -h ] [ -p ] [ -v volume ] [ -V ]

Format  options:  [  -t  filetype  ]  [  -r  rate  ]  [
-s/-u/-U/-A/-a/-g ] [ -b/-w/-l/-f/-d/-D ] [ -c channels  ]
[ -x ]

Effects:
     avg [ -l | -r ]
     band [ -n ] center [ width ]
     check
     chorus  gain-in  gain  out  delay  decay  speed depth
          -s | -t [ delay decay speed depth -s | -fI-t ]
     copy
     cut
     deemph
     echo gain-in gain-out delay decay [ delay decay  ...]
     echos gain-in gain-out delay decay [ delay decay ...]
     flanger gain-in gain-out delay decay speed -s | -fI-t
     highp center
     lowp center
     map
     mask
     phaser gain-in gain-out delay decay speed -s | -t
     pick
     polyphase [ -w < num / ham > ]
               [  -width <  long  / short  / # > ]
               [ -cutoff #  ]
     rate
     resample
     reverb gain-out reverb-time delay [ delay ... ]
     reverse
     split
     stat [ debug | -v ]
     swap [ 1 2 3 4 ]
     vibro speed [ depth ]
```

DESCRIPTION

Sox translates sound files from one format to another, possibly doing a sound effect.

OPTIONS

The option syntax is a little grotty, but in essence:

```
sox file.au file.voc
```

translates a sound sample in SUN Sparc .AU format into a SoundBlaster .VOC file, while

```
sox -v 0.5 file.au -r 12000 file.voc rate
```

does the same format translation but also lowers the amplitude by 1/2 and changes the sampling rate from 8000 hertz to 12000 hertz via the rate sound effect loop.

File type options:

-t filetype
gives the type of the sound sample file.

-r rate Give sample rate in Hertz of file.

-s/-u/-U/-A/-a/-g
The sample data is signed linear (2's complement), unsigned linear, U-law (logarithmic), A-law (logarithmic), ADPCM, or GSM. U-law and A-law are the U.S. and international standards for logarithmic telephone sound compression. ADPCM is form of sound compression that has a good compromise between good sound quality and fast encoding/decoding time. GSM is a standard used for telephone sound compression in European countries and its gaining popularity because of its quality.

-b/-w/-l/-f/-d/-D
The sample data is in bytes, 16-bit words, 32-bit longwords, 32-bit floats, 64-bit double floats, or 80-bit IEEE floats. Floats and double floats are in native machine format.

-x The sample data is in XINU format; that is, it comes from a machine with the opposite word order than yours and must be swapped according to the word-size given above. Only 16-bit and 32-bit integer data may be swapped. Machine-format floating-point data is not portable. IEEE floats are a fixed, portable format. ???

-c channels
The number of sound channels in the data file. This may be 1, 2, or 4; for mono, stereo, or quad sound data.

General options:

-e after the input file allows you to avoid giving an output file and just name an effect. This is mainly useful with the stat effect but can be used with others.

-h Print version number and usage information.

-p Run in preview mode and run fast. This will somewhat speed up sox when the output format has a different number of channels and a different rate then the input file. The order that the effects are run in will be arranged for maximum speed and not quality.

-v volume Change amplitude (floating point); less than 1.0 decreases, greater than 1.0 increases. Note: we perceive volume logarithmically, not linearly.

Note: see the stat effect.

-V Print a description of processing phases. Useful for figuring out exactly how sox is mangling your sound samples.

The input and output files may be standard input and output. This is specified by '-'. The -t type option must be given in this case, else sox will not know the format of the given file. The -t, -r, -s/-u/-U/-A, -b/-w/-l/-f/-d/-D and -x options refer to the input data when given before the input file name. After, they refer to the output data.

If you don't give an output file name, sox will just read the input file. This is useful for validating structured file formats; the stat effect may also be used via the -e option.

FILE TYPES

Sox needs to know the formats of the input and output files. File formats which have headers are checked, if that header doesn't seem right, the program exits with an appropriate message. Currently, raw (no header) binary and textual data, Amiga 8SVX, Apple/SGI AIFF, SPARC .AU (w/header), NeXT .SND, CD-R, CVSD, GSM 06.10, Mac HCOM, Sound Tools MAUD, OSS device drivers, Turtle Beach .SMP, Sound Blaster, Sndtool, and Sounder, Sun Audio device driver, Yamaha TX-16W Sampler, IRCAM Sound Files, Creative Labs VOC, Psion .WVE, and Microsoft RIFF/WAV are supported.

.8svx Amiga 8SVX musical instrument description format.

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

.au SUN Microsystems AU files. There are apparently many types of .au files; DEC has invented its own with a different magic number and word order. The .au handler can read these files but will not write them. Some .au files have valid AU headers and some do not. The latter are probably original SUN u-law 8000 hz samples. These can be dealt with using the .ul format (see below).

.cdr CD-R
CD-R files are used in mastering music Compact

Disks. The file format is, as you might expect, raw stereo raw unsigned samples at 44khz. But, there's some blocking/padding oddity in the format, so it needs its own handler.

.cvs Continuously Variable Slope Delta modulation
Used to compress speech audio for applications such as voice mail.

.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 sample and the sample value. Values are normalized so that the maximum and minimum are 1.00 and -1.00. This file format can be used to create data files for external programs such as FFT analyzers or graph routines. SoX can also convert a file in this format back into one of the other file formats.

.gsm GSM 06.10 Lossy Speech Compression
A standard for compressing speech which is used in the Global Standard for Mobil telecommunications (GSM). Its good for its purpose, shrinking audio data size, but it will introduce lots of noise when a given sound sample 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 your usual arsenal of file converters to deal with an HCOM file under Unix or DOS.

.maud An Amiga format
An IFF-conform sound 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, u-law in mono and stereo.

ossdsp OSS /dev/dsp device driver
This is a psuedo-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 the OSS /dev/dsp file and configure it to use the same

data type as passed in to Sox. It works for both playing and recording sound samples. When playing sound 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 samples your sound card can handle. Example: -t ossdsp -w -s /dev/dsp

.sf IRCAM Sound Files.
SoundFiles are used by academic music software such as the CSound package, and the MixView sound sample editor.

.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.

sunau Sun /dev/audio device driver
This is a psuedo-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 sound samples. When playing sound 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 samples your hardware can handle. Example: -t sunau -w -s /dev/audio or -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 33kHz if the sample rate is still unknown.

.vms More info to come.
Used to compress speech audio for applications such as voice mail.

.voc 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.

.wav Microsoft .WAV RIFF files.
These appear to be very similar to IFF files, but not the same. They are the native sound file format of Windows. (Obviously, Windows was of such incredible importance to the computer industry that it just had to have its own sound file format.) 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. Note that it is possible to write data of a type that cannot be specified by the .wav header, and you will be warned that you a writing a bad file ! Sox currently can read PCM, ULAW, ALAW, MS ADPCM, and IMA (or DVI) ADPCM. It can output all of these formats except the ADPCM styles.

.wve Psion 8-bit alaw
These are 8-bit a-law 8khz sound files used on the Psion palmtop portable computer.

.raw Raw files (no header).
The sample rate, size (byte, word, etc), and style (signed, unsigned, etc.) of the sample file must be given. The number of channels defaults to 1.

.ub, .sb, .uw, .sw, .ul
These are several suffices which serve as a shorthand for raw files with a given size and style. Thus, ub, sb, uw, sw, and ul correspond to "unsigned byte", "signed byte", "unsigned word", "signed word", and "ulaw" (byte). The sample rate defaults to 8000 hz if not explicitly set, and the number of channels (as always) defaults to 1. There are lots of Sparc samples floating around in u-law format with no header and fixed at a sample rate of 8000 hz. (Certain sound management software cheerfully ignores the headers.) Similarly, most Mac sound files are in unsigned byte format with a sample rate of 11025 or 22050 hz.

.auto This is a ‘‘meta-type’’: specifying this type for an input file triggers some code that tries to guess the real type by looking for magic words in the header. If the type can’t be

guessed, the program exits with an error message. The input must be a plain file, not a pipe. This type can't be used for output files.

EFFECTS

Only one effect from the palette may be applied to a sound sample. To do multiple effects you'll need to run sox in a pipeline.

avg [-l | -r]
: Reduce the number of channels by averaging the samples, or duplicate channels to increase the number of channels. Valid combinations are 1 - 2, 1 - 4, 2 - 4, 4 - 2, 4 - 1, 2 - 1. The -l or -r option is not really averaging but either duplicates or leaves just the left or right channel, depending on if your increasing or decreasing the number of output channels.

band [-n] center [width]
: Apply a band-pass filter. The frequency response drops logarithmically around the center frequency. The width 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 signals, i.e. voice, singing, or instrumental music. The -n (for noise) option uses the alternate mode for un-pitched signals. Band introduces noise in the shape of the filter, i.e. peaking at the center frequency and settling around it.

chorus gain-in gain-out delay decay speed deptch

-s | -t [delay decay speed depth -s | -t ...]
: Add a chorus to a sound sample. Each quadtuple 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 sinodial (-s) or triangular (-t). Gain-out is the volume of the output.

copy
: Copy the input file to the output file. This is the default effect if both files have the same sampling rate.

cut loopnumber
: Extract loop #N from a sample.

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

echo gain-in gain-out delay decay [delay decay ...]
Add echoing to a sound sample. Each delay/decay part 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 [delay decay ...]
Add a sequence of echos to a sound sample. Each delay/decay part gives the delay in milliseconds and the decay (relative to gain-in) of that echo. Gain-out is the volume of the output.

flanger gain-in gain-out delay decay speed -s | -t
Add a flanger to a sound sample. Each triple 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 sinodial (-s) or triangular (-t). Gain-out is the volume of the output.

highp center
Apply a high-pass filter. The frequency response drops logarithmically with center frequency in the middle of the drop. The slope of the filter is quite gentle.

lowp center
Apply a low-pass filter. The frequency response drops logarithmically with center frequency in the middle of the drop. The slope of the filter is quite gentle.

map Display a list of loops in a sample, and miscellaneous loop info.

mask Add "masking noise" to signal. This effect deliberately adds white noise to a sound in order to mask quantization effects, created by the process of playing a sound digitally. It tends to mask buzzing voices, for example. It adds 1/2 bit of noise to the sound file at the output bit depth.

phaser gain-in gain-out delay decay speed -s | -t
Add a phaser to a sound sample. Each triple 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 sinodial (-s) or triangular (-t). The decay should be less than 0.5 to avoid feedback. Gain-out is the volume of the output.

pick Select the left or right channel of a stereo sample, or one of four channels in a quadrophonic sample.

polyphase [-w < num / ham >]

[-width < long / short / # >]

[-cutoff #]
Translate input sampling rate to output sampling rate via polyphase interpolation, a DSP algorithm. This method is slow and uses lots of RAM, but gives much better results then rate.
-w < nut / ham > : select either a Nuttal (~90 dB stopband) or Hamming (~43 dB stopband) window. Warning: Nuttall windows require 2x length than Hamming windows. Default is nut.
-width long / short / # : specify the width of the filter. long is 1024 samples; short is 128 samples. Alternatively, an exact number can be used. Default is long.
-cutoff # : specify the filter cutoff frequency in terms of fraction of bandwidth. If upsampling, then this is the fraction of the orignal signal that should go through. If downsampling, this is the fraction of the signal left after downsampling. Default is 0.95. Remember that this is a float.

rate Translate input sampling rate to output sampling rate via linear interpolation to the Least Common Multiple of the two sampling rates. This is the default effect if the two files have different sampling rates and the preview options was specified. This is fast but noisy: the spectrum of the original sound will be shifted upwards and duplicated faintly when up-translating by a multiple. Lerp-ing is acceptable for cheap 8-bit sound hardware, but for CD-quality sound you should instead use either resample or polyphase. If you are wondering which of Sox's rate changing effects to ues, you will want to read a detailed analysis of all of them at http://eakaw2.et.tu-dresden.de/~andreas/resample/resample.html

resample [rolloff [beta]]
Translate input sampling rate to output sampling rate via simulated analog filtration. This method is slower than rate, but gives much better results. 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 therefor should be something between 0. and 1., in practice 0.8-0.95. beta trades stop band rejection against transition width from passband to stop band. Larger beta means a slower transition and greater stopband rejection. beta should be at least greater than 2. The default is rollof 0.8, beta 17.5, which is rather conservative with respect to

aliasing. Lower beta and higher rolloff values preserve more high frequency signal energy, but introduce measurable artifacts. This is the default effect if the two files have different sampling rates.

reverb gain-out delay [delay ...]
Add reverbation to a sound sample. 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 reverbation. Gain-out is the volume of the output.

reverse Reverse the sound sample completely. Included for finding Satanic subliminals.

split Turn a mono sample into a stereo sample by copying the input channel to the left and right channels.

stat [debug | -v]
Do a statistical check on the input file, and print results on the standard error file. stat may copy the file untouched from input to output, if you select an output file. The "Volume Adjustment:" field in the statistics gives you the argument to the -v number which will make the sample as loud as possible without clipping. There is an optional parameter -v that will print out the "Volume Adjustment:" field's value and return. This could be of use in scripts to auto convert the volume. There is an also an optional parameter debug that will place sox into debug mode and print out a hex dump of the sound file 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.

swap [1 2 3 4]
Swap channels in multi-channel sound files. In files with more than 2 channels you may specify the order that the channels should be rearranged in.

vibro speed [depth]
Add the world-famous Fender Vibro-Champ sound effect to a sound sample by using a sine wave as the volume knob. Speed gives the Hertz value of the wave. This must be under 30. Depth gives the amount the volume is cut into by the sine wave, ranging 0.0 to 1.0 and defaulting to 0.5.

Sox enforces certain effects. If the two files have different sampling rates, the requested effect must be one of copy, or rate, If the two files have different numbers of

channels, the avg effect must be requested.

BUGS

The syntax is horrific. It's very tempting to include a default system that allows an effect name as the program name and just pipes a sound sample from standard input to standard output, but the problem of inputting the sample rates makes this unworkable.

Please report any bugs found in this version of sox to Chris Bagwell (cbagwell@sprynet.com)

FILES

SEE ALSO

play(1), rec(1)

NOTICES

The version of Sox that accompanies this manual page is support by Chris Bagwell (cbagwell@sprynet.com). Please refer any questions regarding it to this address. You may obtain the latest version at the the web site http://home.sprynet.com/~cbagwell/sox.html