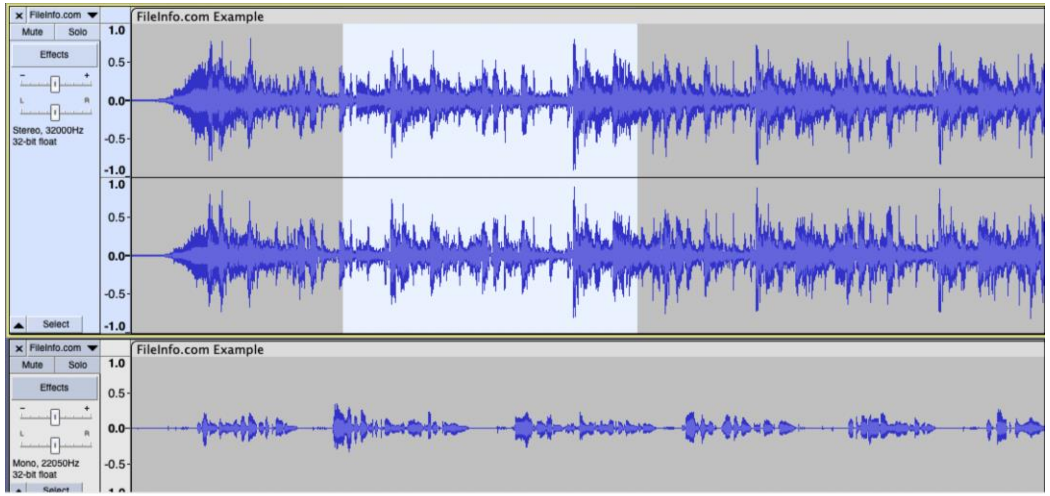
RV College of Engineering®

Go, change the world

Linear Integrated Circuits and Applications – Experiential Learning Phase - I



TOPIC:

Programmatic Simulation of Digital Filters for Audio Applications

Team Introduction

Roll No.	USN	Name	Email Id
1.	1RV22ET005	ARUN K R	arunkr.et22@rvce.edu.in
2.	1RV22ET062	YASHAS KASHYAP	yashaskashyap.et22@rvce.edu.in
3.	RVCE23BTE401	S R SREEJITH	srsreejith.et23@rvce.edu.in
4.	1RV22ET016	CHIRANTHAN BHARADVAJ B	chiranthanbb.et22@rvce.edu.in

Introduction

- Sound eXchange (SoX) is an open source cross-platform audio editing software. It has a command line interface and is written in ‘C’.

- Some of SoX’s features include:
  1) Reading and writing into various audio file formats like mp3, wav, aiff etc.
  2) Recording and playing audio.
  3) Editing via concatenate, trim , pad, repeat, reverse, volume, fade etc.
  4) Noise removal using frequency profiling.
  5) Multi-track mixing
  6) Spectrogram analysis
  7) Adjustment of speed (pitch and tempo) and sample rate.

- Bash is a powerful tool for automating system administration tasks and performing other routine tasks in Unix/Linux.

- Sonic Visualiser and Audacity are two audio editing softwares.

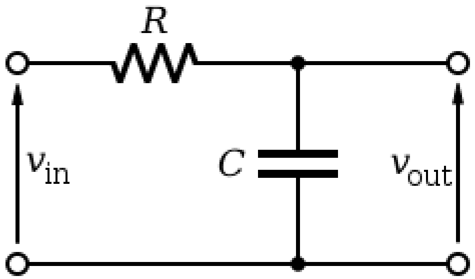
Objectives

- Using the "Libsox" library to design digital filter like low-pass, high-pass, band pass and band reject.
- Creating multiple files using "bash" scripting and remixing them using "sox" commands.
- Analysing spectrograms of the mixed frequency files using the tool "Sonic Visualiser".
- Noise is any unwanted signals in the audio and can make it unpleasant to hear. For thisf reason, many youtubers do post audio processing to remove any noise present in their audio. We will be using our filters to remove unwanted noises from different audio files.
- To compare input and output files on how they sound before and after they pass through various filters.
- Analysing spectrograms of files to visualise the filters' response.

Low-pass Filter

- **A Low-pass Filter is a filter that is designed to pass all frequencies below a given cut-off frequency. All frequencies higher than the cut-off frequency are attenuated.**

Simple first order low-pass RC filter:-



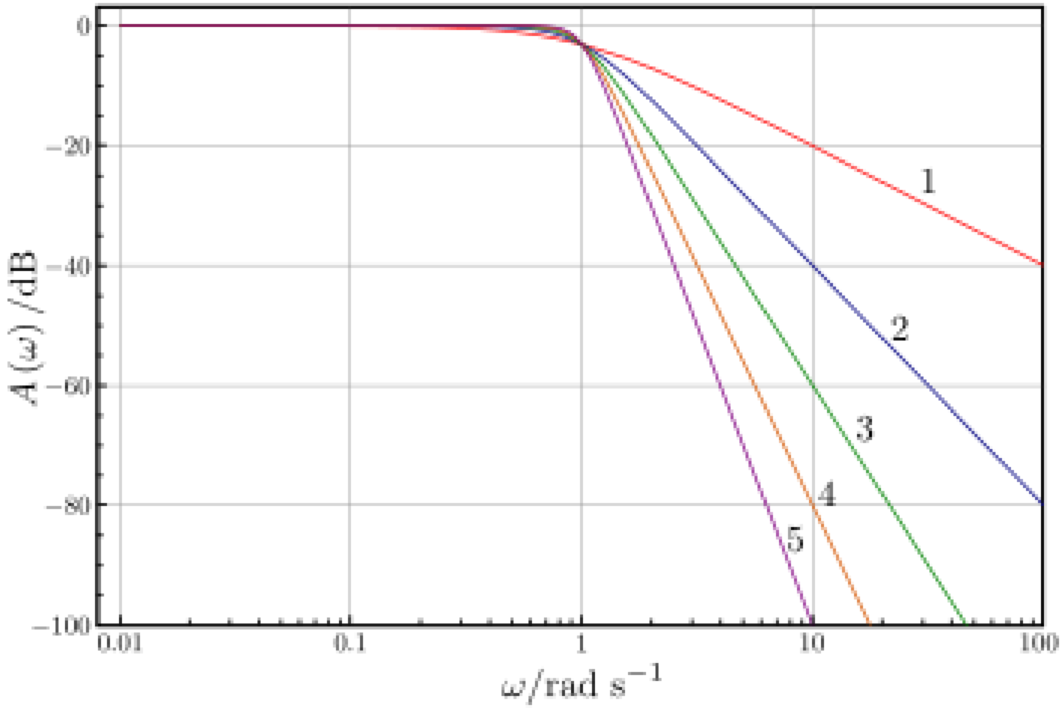
Transfer function:-

$$H(s) = \frac{V_{\text{out}}(s)}{V_{\text{in}}(s)} = \frac{\omega_0}{(s+\omega_0)}$$

Cut-off Frequency:-

$$f_c = \frac{1}{2\pi\tau} = \frac{1}{2\pi RC},$$

Frequency Response:-

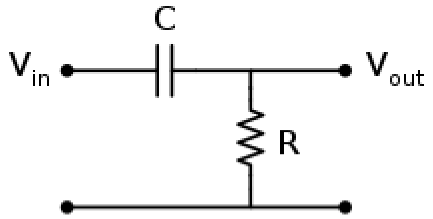


Roll-off = -20n dB/decade where n is the filter order

High-pass Filter

- A High-pass Filter is a filter that is designed to pass all frequencies above a given cut-off frequency. All frequencies lower than the cut-off frequency are attenuated.

Simple first order high- pass filter



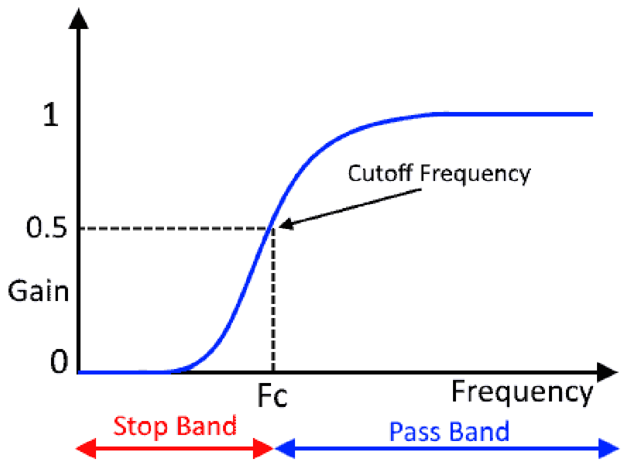
Transfer Function

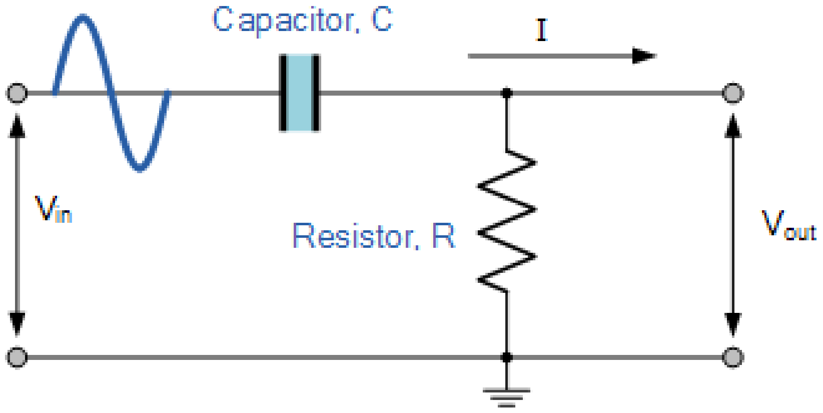
$$\frac{V_{\text{out}}(s)}{V_{\text{in}}(s)} = \frac{sRC}{1+sRC}.$$

Cut-off Frequency

$$f_c = \frac{1}{2\pi\tau} = \frac{1}{2\pi RC},$$

Frequency Response



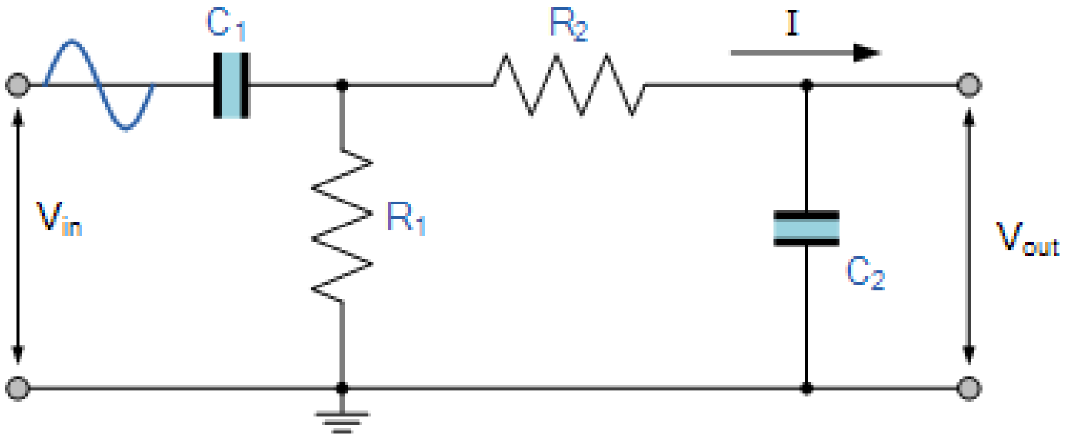


Band-pass Filter

- ➢ **A Band-pass Filter is a filter that is designed to pass all frequencies that fall between its cut-off frequencies fc1 and fc2. All frequencies outside the range are attenuated.**

- ➢ **Can be created by combining low-pass and high-pass filters in series.**

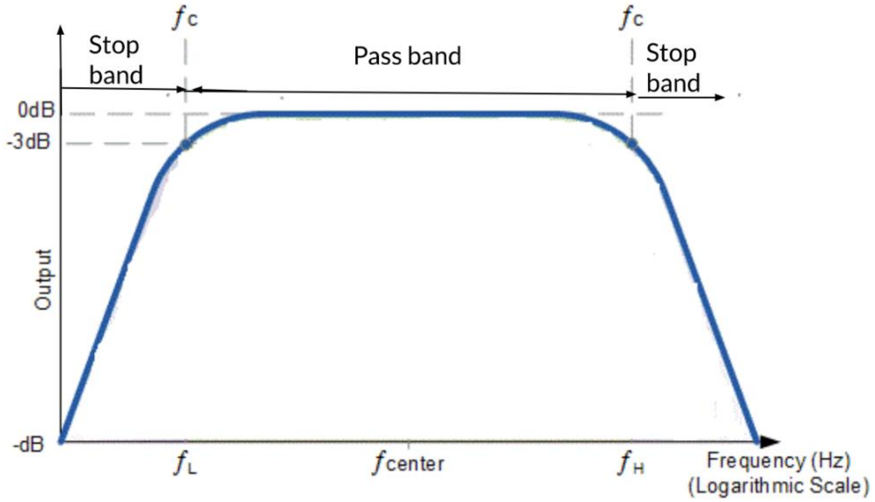
Simple band-pass filter circuit:-



High pass filter cutoff frequency: $\mathbf{fc1} = \dfrac{1}{2\pi R1C1}$

Low-pass filter cutoff frequency: $\mathbf{fc2} = \dfrac{1}{2\pi R2C2}$

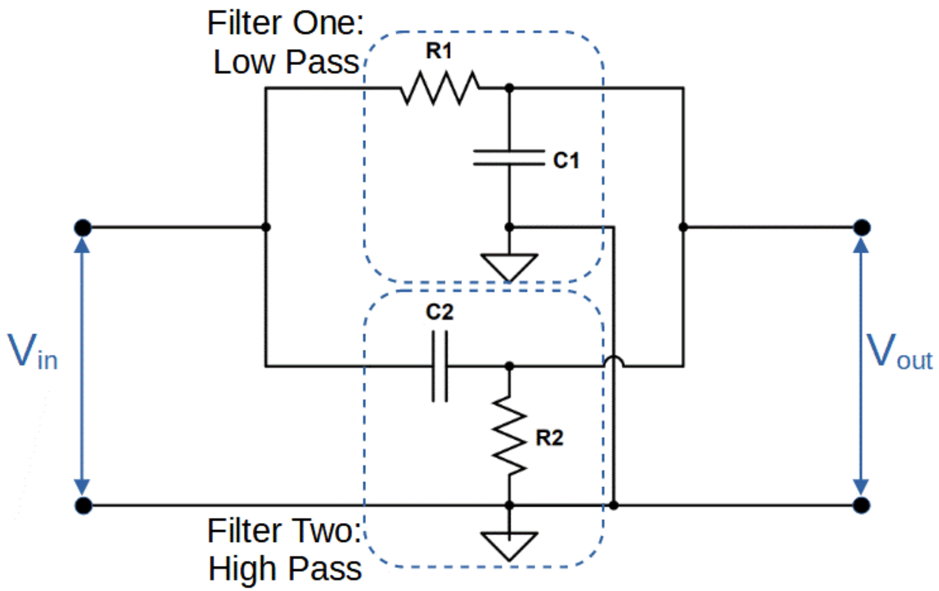
Frequency Response:-



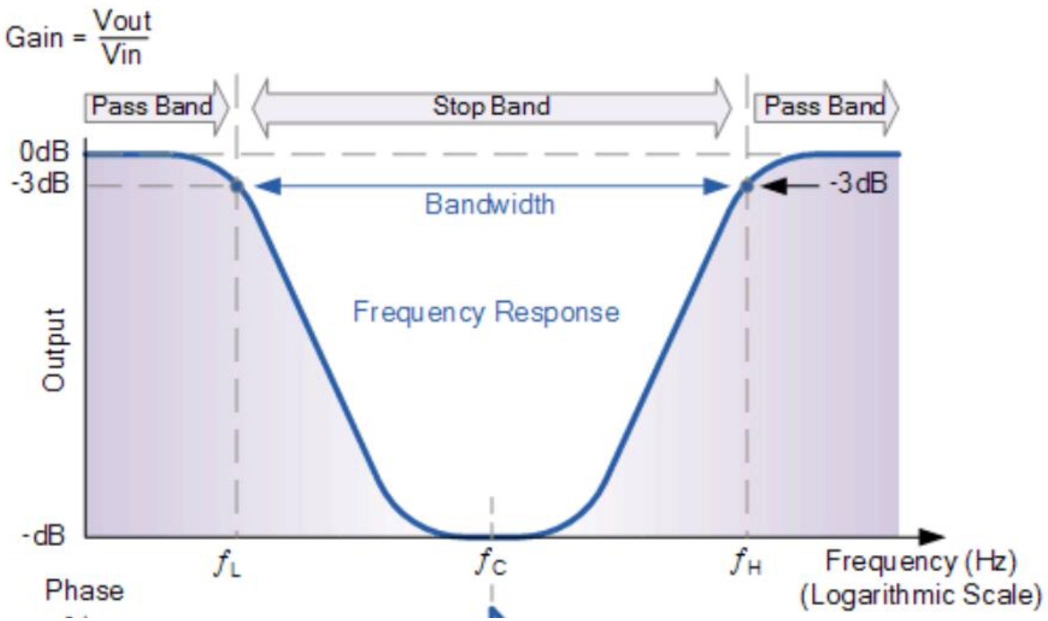
Band-stop Filter

- ➢ A Band-pass Filter is a filter that is designed to block all frequencies that fall between its cut-off frequencies fc1 and fc2. All frequencies outside the range are allowed to pass through.
- ➢ Can be created by combining low-pass and high-pass filters in parallel.

Simple band-stop filter circuit:-



Frequency Response:-

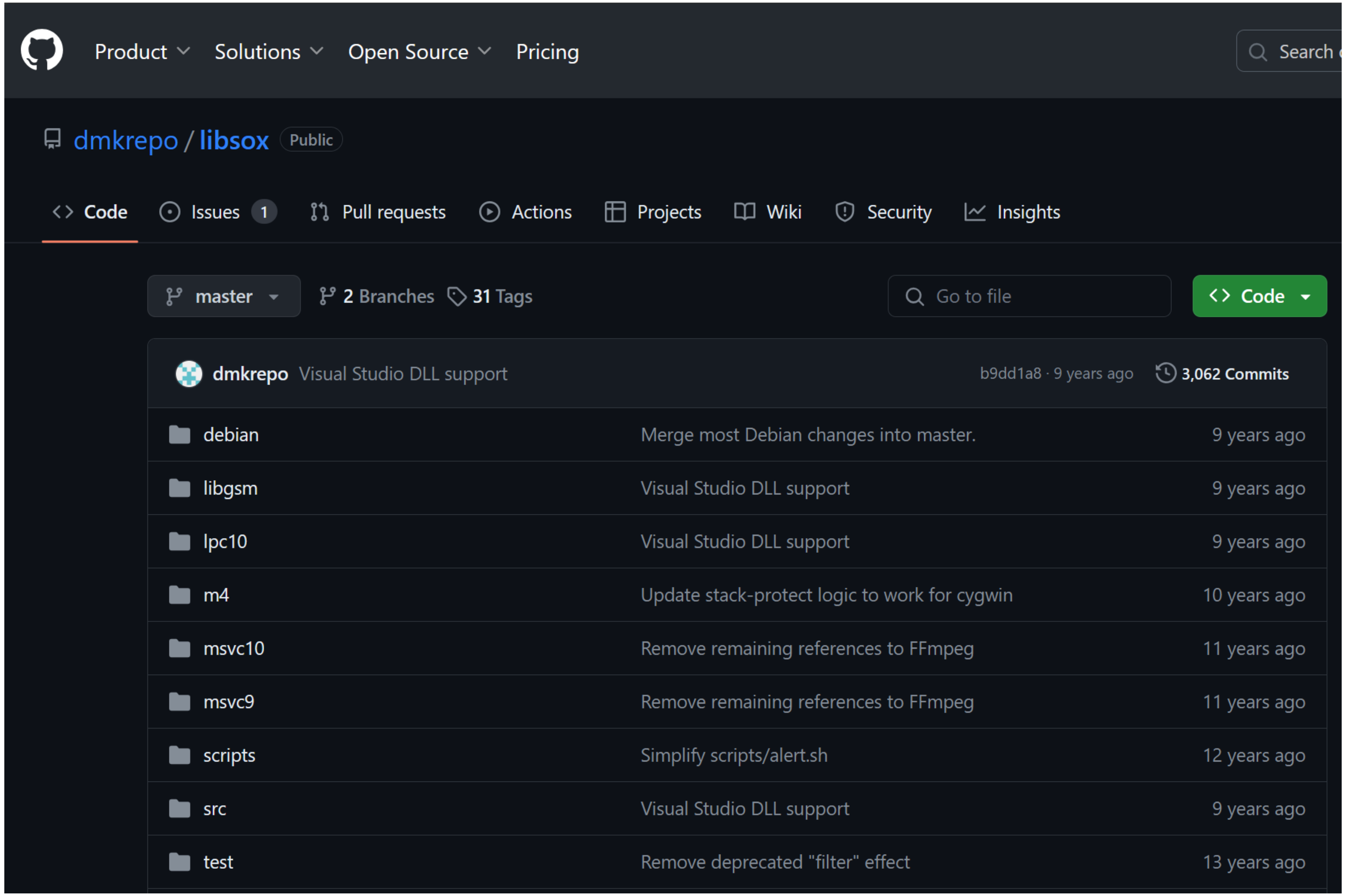


Execution in Phase-2

- **Implementing the open source library called "LibSoX". Its Source Code is available on Github.**
- **Creating our own audio files (at least 100 files) using 'sox' command and bash scripting.**
- **Mixing multiple files to create a mixed frequency audio files. The files vary in parameters like volume, type of waveform (sin, sawtooth), no. of channels etc.**
- **Writing code for digital filters like low pass, high pass, band pass and band reject filters.**
- **Passing mixed frequency files as input to these digital filters and storing output in separate files.**
- **Analysing frequency spectrum of these output files on software tool "Sonic Visualizer" and "Audacity".**

Execution in Phase-2

GITHUB REPOSITORY OF “LIBSOX” :-



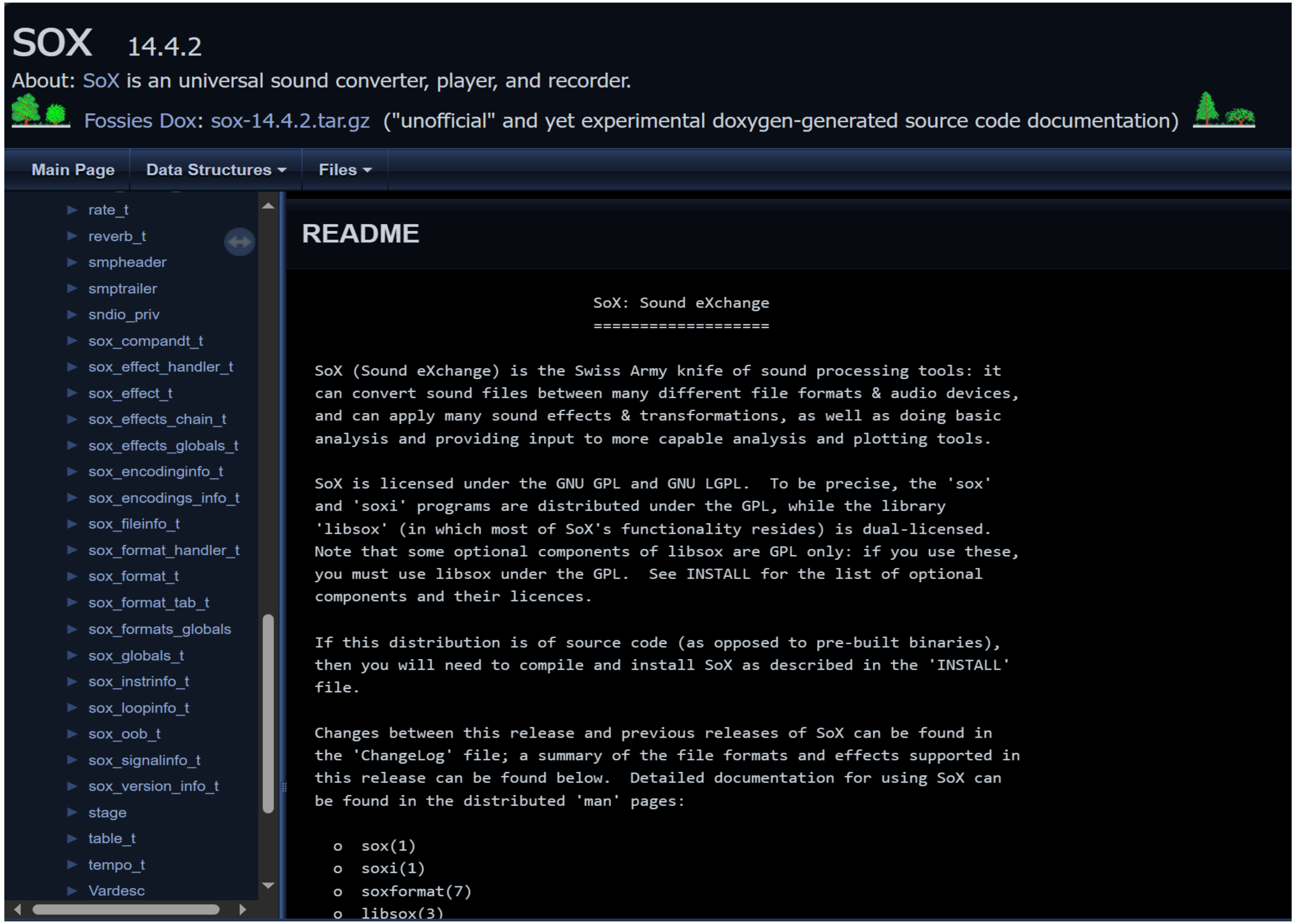
dmkrepo / libsox

dmkrepo Visual Studio DLL support b9dd1a8 · 9 years ago 3,062 Commits

debian	Merge most Debian changes into master.	9 years ago
libgsm	Visual Studio DLL support	9 years ago
lpc10	Visual Studio DLL support	9 years ago
m4	Update stack-protect logic to work for cygwin	10 years ago
msvc10	Remove remaining references to FFmpeg	11 years ago
msvc9	Remove remaining references to FFmpeg	11 years ago
scripts	Simplify scripts/alert.sh	12 years ago
src	Visual Studio DLL support	9 years ago
test	Remove deprecated "filter" effect	13 years ago

Execution in Phase-2

DOCUMENTATION OF “LIBSOX” LIBRARY :-



SOX 14.4.2

About: SoX is an universal sound converter, player, and recorder.

Fossies Dox: sox-14.4.2.tar.gz ("unofficial" and yet experimental doxygen-generated source code documentation)

README

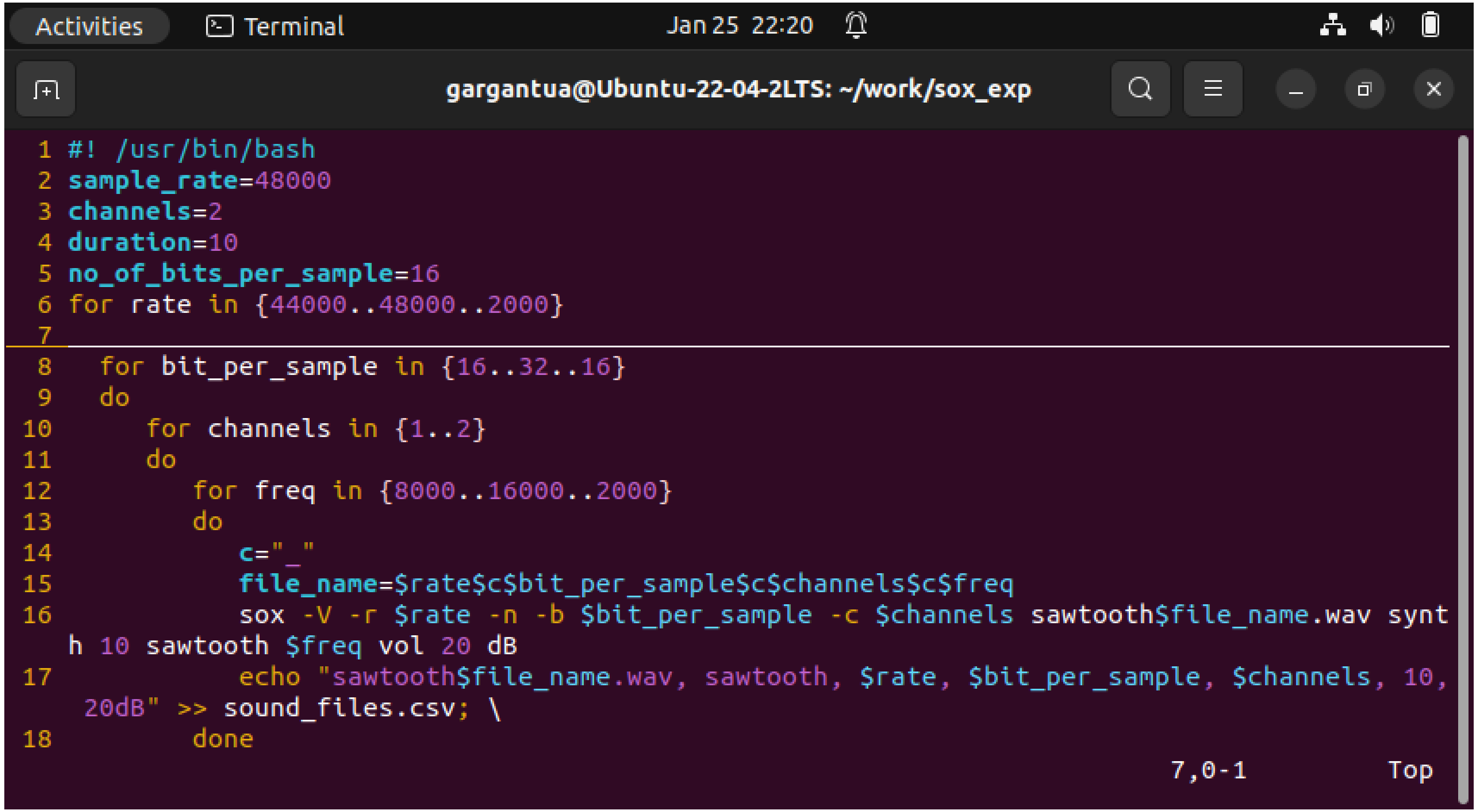
```
                              SoX: Sound eXchange
                              ===================

SoX (Sound eXchange) is the Swiss Army knife of sound processing tools: it
can convert sound files between many different file formats & audio devices,
and can apply many sound effects & transformations, as well as doing basic
analysis and providing input to more capable analysis and plotting tools.

SoX is licensed under the GNU GPL and GNU LGPL.  To be precise, the 'sox'
and 'soxi' programs are distributed under the GPL, while the library
'libsox' (in which most of SoX's functionality resides) is dual-licensed.
Note that some optional components of libsox are GPL only: if you use these,
you must use libsox under the GPL.  See INSTALL for the list of optional
components and their licences.

If this distribution is of source code (as opposed to pre-built binaries),
then you will need to compile and install SoX as described in the 'INSTALL'
file.

Changes between this release and previous releases of SoX can be found in
the 'ChangeLog' file; a summary of the file formats and effects supported in
this release can be found below.  Detailed documentation for using SoX can
be found in the distributed 'man' pages:

  o  sox(1)
  o  soxi(1)
  o  soxformat(7)
  o  libsox(3)
```

RV College of Engineering®

Execution in Phase-2

Go, change the world

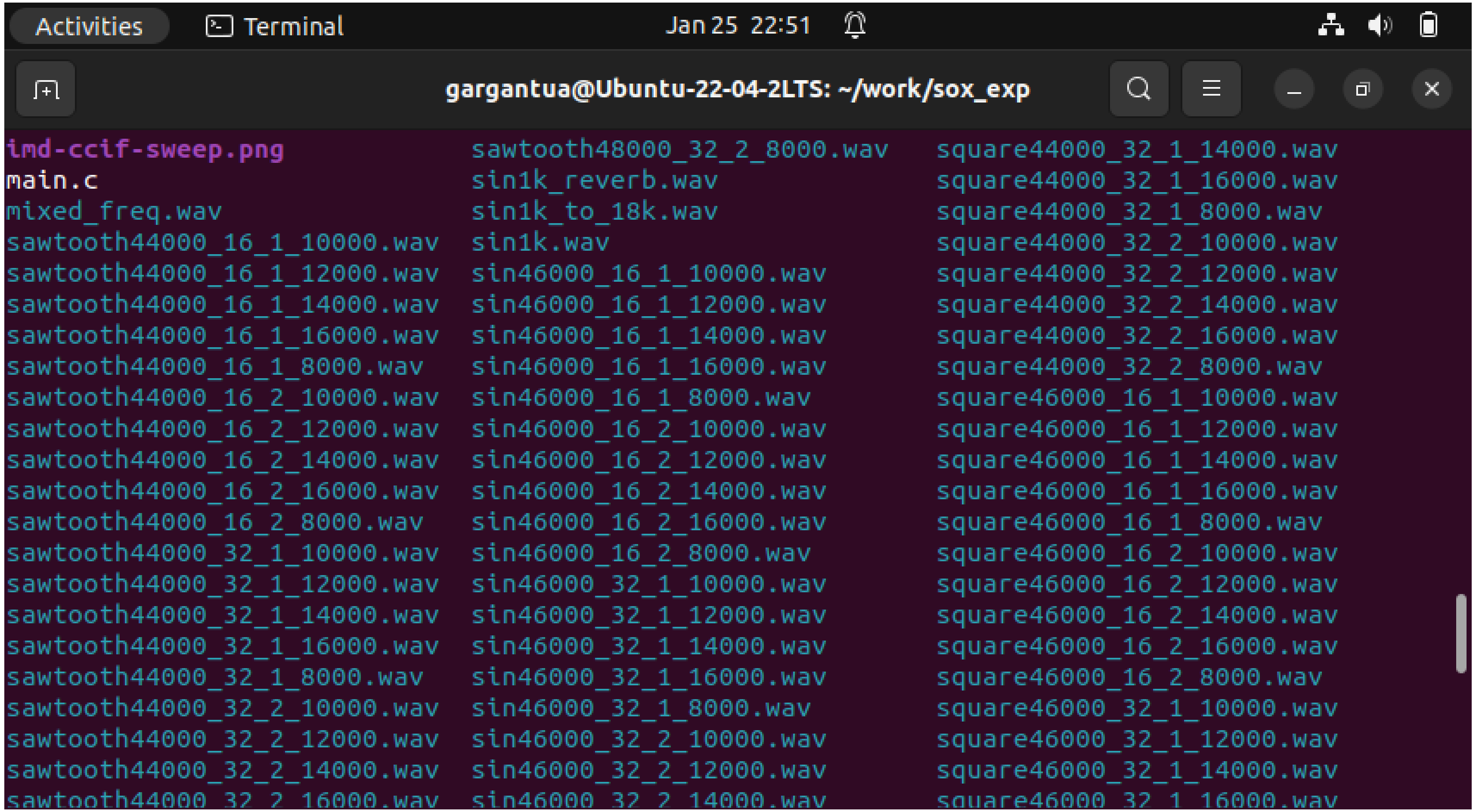
BASH SCRIPT FOR CREATING MULTIPLE FILES :-



```
#! /usr/bin/bash
sample_rate=48000
channels=2
duration=10
no_of_bits_per_sample=16
for rate in {44000..48000..2000}

  for bit_per_sample in {16..32..16}
  do
    for channels in {1..2}
    do
      for freq in {8000..16000..2000}
      do
        c="_"
        file_name=$rate$c$bit_per_sample$c$channels$c$freq
        sox -V -r $rate -n -b $bit_per_sample -c $channels sawtooth$file_name.wav synth 10 sawtooth $freq vol 20 dB
        echo "sawtooth$file_name.wav, sawtooth, $rate, $bit_per_sample, $channels, 10, 20dB" >> sound_files.csv; \
      done
```

Execution in Phase-2

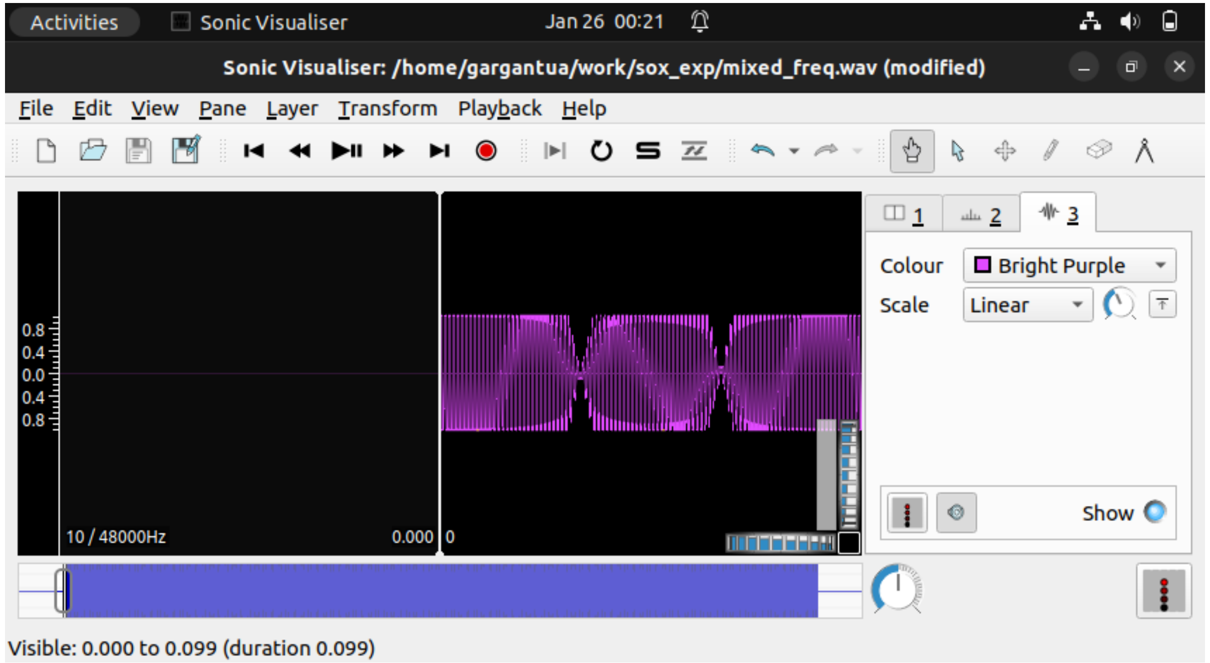
AFTER RUNNING THE BASH SCRIPT :-



```
imd-ccif-sweep.png            sawtooth48000_32_2_8000.wav   square44000_32_1_14000.wav
main.c                        sin1k_reverb.wav              square44000_32_1_16000.wav
mixed_freq.wav                sin1k_to_18k.wav              square44000_32_1_8000.wav
sawtooth44000_16_1_10000.wav  sin1k.wav                     square44000_32_2_10000.wav
sawtooth44000_16_1_12000.wav  sin46000_16_1_10000.wav       square44000_32_2_12000.wav
sawtooth44000_16_1_14000.wav  sin46000_16_1_12000.wav       square44000_32_2_14000.wav
sawtooth44000_16_1_16000.wav  sin46000_16_1_14000.wav       square44000_32_2_16000.wav
sawtooth44000_16_1_8000.wav   sin46000_16_1_16000.wav       square44000_32_2_8000.wav
sawtooth44000_16_2_10000.wav  sin46000_16_1_8000.wav        square46000_16_1_10000.wav
sawtooth44000_16_2_12000.wav  sin46000_16_2_10000.wav       square46000_16_1_12000.wav
sawtooth44000_16_2_14000.wav  sin46000_16_2_12000.wav       square46000_16_1_14000.wav
sawtooth44000_16_2_16000.wav  sin46000_16_2_14000.wav       square46000_16_1_16000.wav
sawtooth44000_16_2_8000.wav   sin46000_16_2_16000.wav       square46000_16_1_8000.wav
sawtooth44000_32_1_10000.wav  sin46000_16_2_8000.wav        square46000_16_2_10000.wav
sawtooth44000_32_1_12000.wav  sin46000_32_1_10000.wav       square46000_16_2_12000.wav
sawtooth44000_32_1_14000.wav  sin46000_32_1_12000.wav       square46000_16_2_14000.wav
sawtooth44000_32_1_16000.wav  sin46000_32_1_14000.wav       square46000_16_2_16000.wav
sawtooth44000_32_1_8000.wav   sin46000_32_1_16000.wav       square46000_16_2_8000.wav
sawtooth44000_32_2_10000.wav  sin46000_32_1_8000.wav        square46000_32_1_10000.wav
sawtooth44000_32_2_12000.wav  sin46000_32_2_10000.wav       square46000_32_1_12000.wav
sawtooth44000_32_2_14000.wav  sin46000_32_2_12000.wav       square46000_32_1_14000.wav
sawtooth44000_32_2_16000.wav  sin46000_32_2_14000.wav       square46000_32_1_16000.wav
```
]]>

Execution in Phase-2

REMIXING THE .WAV FILES, VISUALISING WAVEFORM AND SPECTOGRAM:-



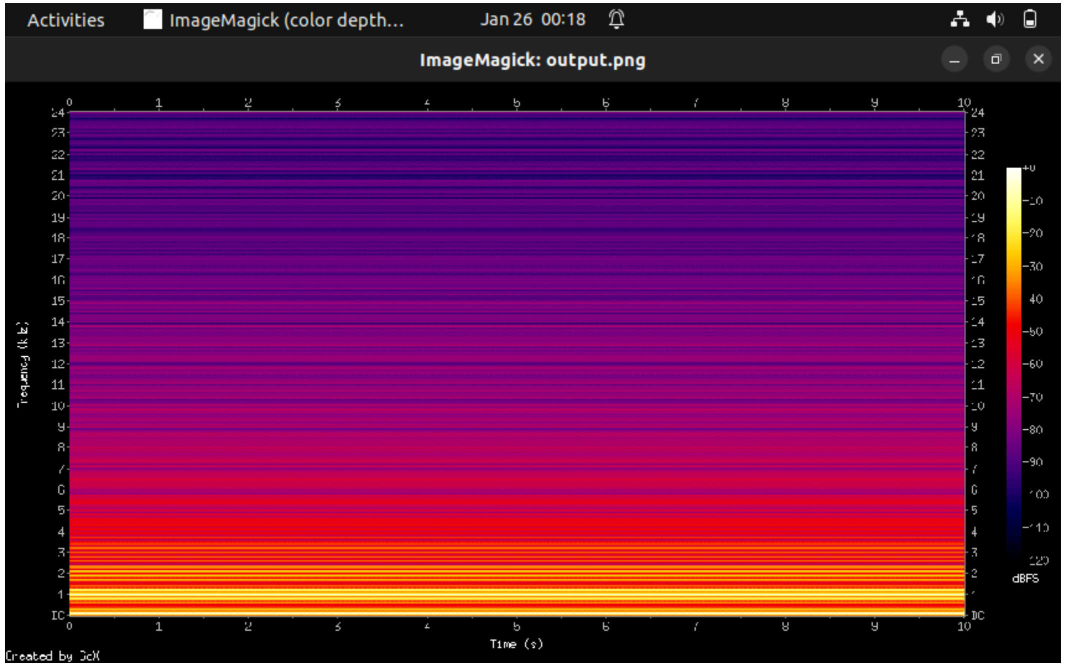
Sonic Visualiser: /home/gargantua/work/sox_exp/mixed_freq.wav (modified)

Colour: Bright Purple

Scale: Linear

10 / 48000Hz

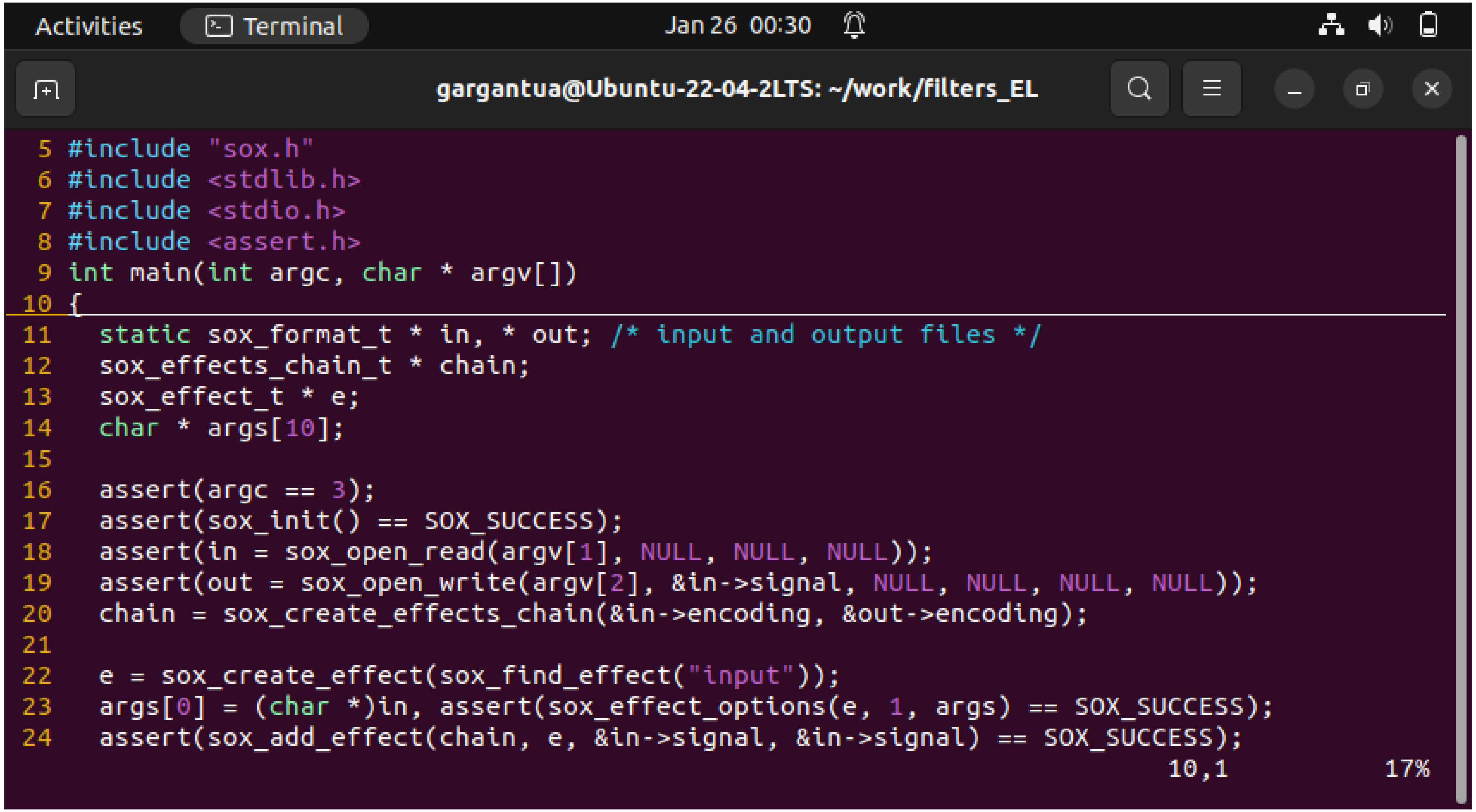
Visible: 0.000 to 0.099 (duration 0.099)



ImageMagick: output.png

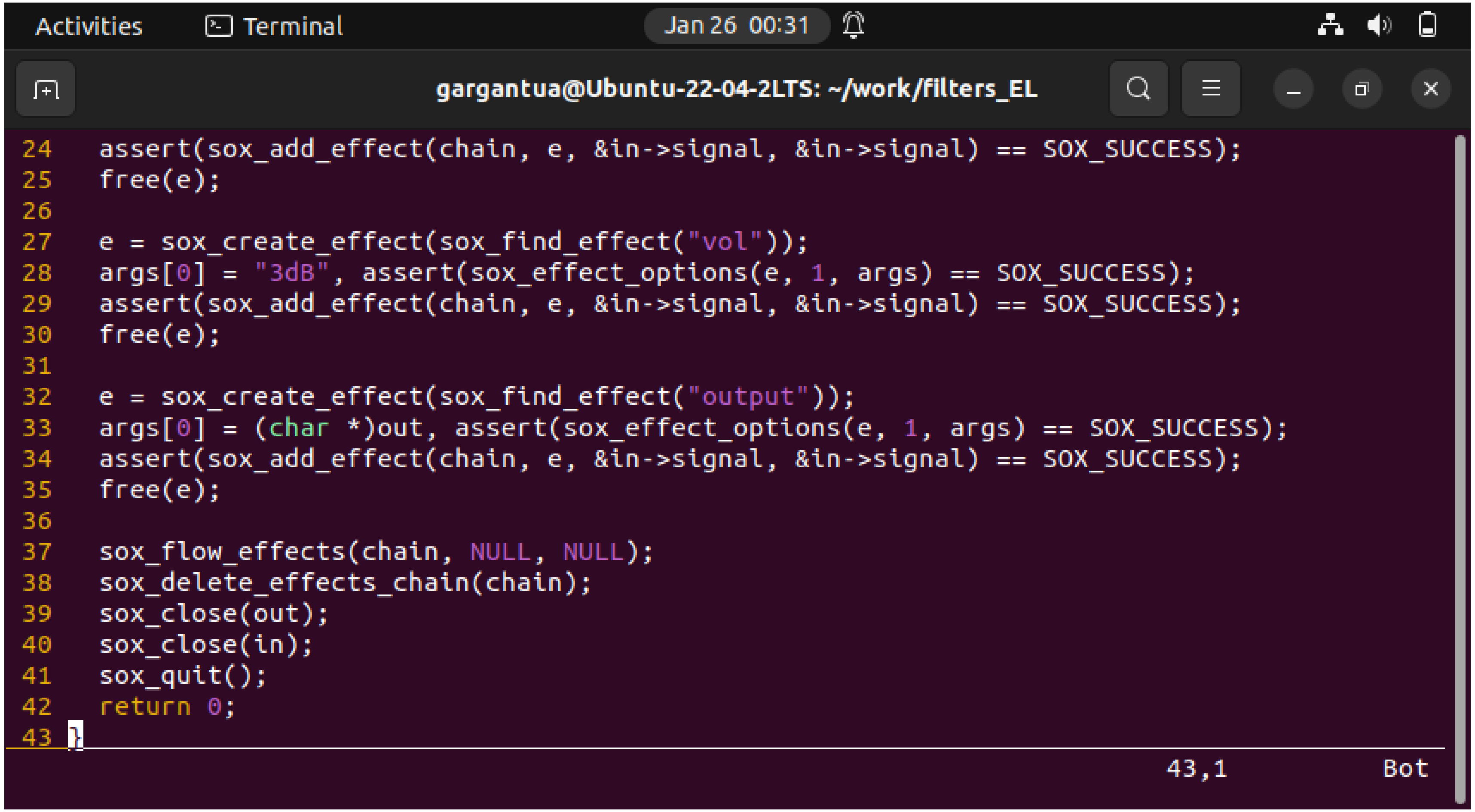
Execution in Phase-2

SAMPLE CODE FOR VOLUME EFFECT:-



```
#include "sox.h"
#include <stdlib.h>
#include <stdio.h>
#include <assert.h>
int main(int argc, char * argv[])
{
  static sox_format_t * in, * out; /* input and output files */
  sox_effects_chain_t * chain;
  sox_effect_t * e;
  char * args[10];

  assert(argc == 3);
  assert(sox_init() == SOX_SUCCESS);
  assert(in = sox_open_read(argv[1], NULL, NULL, NULL));
  assert(out = sox_open_write(argv[2], &in->signal, NULL, NULL, NULL, NULL));
  chain = sox_create_effects_chain(&in->encoding, &out->encoding);

  e = sox_create_effect(sox_find_effect("input"));
  args[0] = (char *)in, assert(sox_effect_options(e, 1, args) == SOX_SUCCESS);
  assert(sox_add_effect(chain, e, &in->signal, &in->signal) == SOX_SUCCESS);
                                                                  10,1          17%
```

Execution in Phase-2



```
  assert(sox_add_effect(chain, e, &in->signal, &in->signal) == SOX_SUCCESS);
  free(e);

  e = sox_create_effect(sox_find_effect("vol"));
  args[0] = "3dB", assert(sox_effect_options(e, 1, args) == SOX_SUCCESS);
  assert(sox_add_effect(chain, e, &in->signal, &in->signal) == SOX_SUCCESS);
  free(e);

  e = sox_create_effect(sox_find_effect("output"));
  args[0] = (char *)out, assert(sox_effect_options(e, 1, args) == SOX_SUCCESS);
  assert(sox_add_effect(chain, e, &in->signal, &in->signal) == SOX_SUCCESS);
  free(e);

  sox_flow_effects(chain, NULL, NULL);
  sox_delete_effects_chain(chain);
  sox_close(out);
  sox_close(in);
  sox_quit();
  return 0;
}
```

Applications

LOW-PASS:-

- Used to remove high frequency noise from biomedical signals, such as ECG signals ensuring accurate and reliable monitoring of physiological parameters.
- Often used in power supply circuits to filter out high frequency noise to provide a stable and clean DC output voltage.

HIGH-PASS:-

- In image-processing, high pass filters are used to detect edges in images. It allows the higher frequency components associated with the edges to pass through. Moreover, using high pass filters can be used to make an image look more sharper.
- In geophysics, high pass filters are used in seismic signal processing to remove low frequency noise and focus on high frequency seismic signals

BAND-PASS:-

- Used in speech processing applications to focus on specific frequency ranges associated with human speech. This is important in applications like voice recognition.
- Help in channelizing and separating signals to avoid interference in wireless communication systems.

References

- **GITHUB REPOSITORY OF “LIBSOX” LIBRARY:** https://github.com/dmkrepo/libsox
- **SOX LIBRARY DOCUMENTATION:** https://fossies.org/dox/sox-14.4.2/libsox_8c.html
- **MAN PAGES (LINUX) OF “LIBSOX”:** https://linux.die.net/man/3/libsox
- **LIBSOX TUTORIAL:** https://www.audiosciencereview.com/forum/index.php?threads/howto-sox-audio-tool-as-a-signal-generator.4242/

Thank You!