Documentation: Creating and Using Audio Filters in VLC

This document provides a comprehensive guide for both users and developers on working with audio filters in VLC media player. It covers two main topics:

1. **Using FFmpeg avfilters in VLC**: The easy and powerful way to apply any of FFmpeg's hundreds of audio filters directly within VLC without any programming.
2. **Creating a Native VLC Audio Filter**: An advanced guide for C developers on how to build a custom audio filter from source.

1. Using FFmpeg Filters in VLC (The Generic avfilter Plugin)

VLC includes a powerful, generic audio filter module named `avfilter` which acts as a bridge to FFmpeg's `libavfilter` library. This allows you to use almost any FFmpeg audio filter by simply typing its name and parameters.

How It Works

The avfilter plugin in VLC takes a filter-chain string, exactly like the one you would use with the `-af` or `-filter`: a option in the FFmpeg command line. VLC then passes this chain to the FFmpeg library, which processes the audio in real-time.

A. Enabling avfilter via the GUI (Graphical Interface)

This is the most straightforward method for experimentation.

1. **Open VLC Preferences**: Go to Tools -> Preferences.
2. **Show All Settings**: In the bottom-left corner, select the "All" radio button to display advanced settings.
3. **Navigate to the FFmpeg Filter**: In the settings tree on the left, navigate to `Audio -> Filters -> FFmpeg` audio filter.
4. **Enter Your Filter Chain**: In the text box labeled **"FFmpeg audio filter"**, enter the filter graph you want to apply.
5. **Enable the Filter**: Now, navigate back up to `Audio -> Filters` in the tree. Check the box for **"FFmpeg audio filter"** to enable it.
6. **Save and Restart**: Click "Save" and restart VLC for the changes to take effect.

Filter Chain Examples for the GUI

You enter just the filter description, not the ffmpeg `-af` part.

- **Dynamic Audio Normalizer (`dynaudnorm`)**: To apply the filter from our previous discussion.

```
dynaudnorm=f=150:g=31:p=0.95
```

- **Audio Compressor (compressor)**: Great for "night mode" listening.

```
compressor=threshold=0.1:ratio=4:attack=5:release=50
```

- **10-Band Equalizer (equalizer)**: To boost bass and treble (frequencies are in Hz).

```
equalizer=f=60:width_type=h:width=20:g=5,
equalizer=f=120:width_type=h:width=40:g=3,
equalizer=f=8000:width_type=h:width=1000:g=4,
equalizer=f=16000:width_type=h:width=2000:g=6
```

B. Enabling `avfilter` via the Command Line

For scripting or power users, the command line is more efficient. You use the --audio-filter option to enable the module and `--avfilter-graph` to provide the filter chain.

```
# Example using dynaudnorm
vlc "my_movie.mkv" --audio-filter=avfilter
--avfilter-graph="dynaudnorm=p=0.95:g=21"

# Example chaining a highpass and lowpass filter to isolate mid-range
frequencies
vlc "my_podcast.mp3" --audio-filter=avfilter
--avfilter-graph="highpass=f=300,lowpass=f=3000"
```

2. Creating a Native VLC Audio Filter (For C Developers)

If the vast library of FFmpeg filters isn't enough and you need to implement a custom algorithm, you can write a native VLC audio filter. This requires a C development environment and the VLC source code. Given your preference for C++17 with gcc-9.2, you are well-equipped to handle the C-based compilation process.

A. Prerequisites

1. **VLC Source Code**: Clone the official repository: git clone `https://code.videolan.org/videolan/vlc.git`
2. **Build Environment**: A C compiler (like GCC), make, autoconf, and other standard build tools.
3. **Dependencies**: You must install all of VLC's build dependencies for your system. The VLC wiki has excellent guides for Linux, Windows, and macOS.

B. The Core Concepts

A VLC audio filter is a shared library (.so, .dll, .dylib) that conforms to the VLC module API. It primarily consists of:

- **Entry/Exit Points**: Macros (`vlc_module_begin`, `vlc_module_end`) that define your module's properties and capabilities.
- **Callback Functions**:
  - `Open`: Called when the filter is first needed. You allocate memory and initialize your filter's state here.
  - `Filter`: The core function. It's called for every chunk of audio data. This is where your processing logic goes.
  - `Flush`: Called when there is a discontinuity in the stream (e.g., seeking). You should reset your filter's state here.
  - `Close`: Called when the stream ends. You free any memory you allocated in Open.

C. Example: A Simple "Gain" Filter

Let's create a filter that simply multiplies the audio samples by a constant factor to change the volume.

1. Create the File:
Create a new C file inside the VLC source tree at `vlc/modules/audio_filter/gain.c`.

2. Write the Code (`gain.c`):

```
#ifdef HAVE_CONFIG_H
# include "config.h"
#endif

#include <vlc_common.h>
#include <vlc_plugin.h>
```

```
#include <vlc_filter.h>
#include <vlc_block.h>

/*
 * Module definitions
 */
static int  Open(vlc_object_t *);
static void Close(vlc_object_t *);
static block_t *Filter(filter_t *, block_t *);

vlc_module_begin ()
    set_shortname("Gain")
    set_description("Simple audio gain filter")
    set_capability("audio filter", 0)
    set_callbacks(Open, Close)
    add_shortcut("gain")
vlc_module_end ()

/*
 * The filter's internal data structure
 */
struct filter_sys_t
{
    float gain_factor;
};

/*
 * Open: Called to create the filter.
 */
static int Open(vlc_object_t *p_this)
{
    filter_t *p_filter = (filter_t *)p_this;
    filter_sys_t *p_sys;

    /* Allocate our private data structure */
    p_sys = malloc(sizeof(filter_sys_t));
    if (p_sys == NULL)
        return VLC_ENOMEM;

    /* Set a default gain factor (e.g., 1.5 for a 50% boost) */
    p_sys->gain_factor = 1.5f;
    msg_Info(p_filter, "Simple Gain filter initialized with factor: %.2f",
p_sys->gain_factor);

    p_filter->p_sys = p_sys;
    p_filter->pf_audio_filter = Filter;

    return VLC_SUCCESS;
}

/*
 * Close: Called to destroy the filter.
 */
static void Close(vlc_object_t *p_this)
{
    filter_t *p_filter = (filter_t *)p_this;
    filter_sys_t *p_sys = p_filter->p_sys;

    free(p_sys);
}

/*
 * Filter: The core processing function.
 */
static block_t *Filter(filter_t *p_filter, block_t *p_block)
{
    filter_sys_t *p_sys = p_filter->p_sys;

    // Ensure we have an audio block to process
    if (!p_block || !p_block->i_nb_samples ||
p_filter->fmt_in.audio.i_format != VLC_CODEC_FL32)
    {
        // For simplicity, this filter only works on Float32 samples.
        // A real-world filter should handle format conversion.
        if(p_block)
            msg_Warn(p_filter, "Can only process 32-bit float audio.");
        return p_block;
    }

    // Iterate through all samples in the block
    // The block contains interleaved audio data (L, R, L, R, ...)
    float *p_samples = (float *)p_block->p_buffer;
    for (size_t i = 0; i < p_block->i_nb_samples *
p_filter->fmt_in.audio.i_channels; i++)
```

```
    {
        // Apply the gain factor
        p_samples[i] = p_samples[i] * p_sys->gain_factor;
    }

    return p_block;
}
```

D. Building and Installing the Filter

1. **Add to Build System**: Open `vlc/modules/audio_filter/Makefile.am` and add `gain.c` to the `SOURCES_gain` list. Since there is no gain module yet, you will need to add a new section:

```
# In modules/audio_filter/Makefile.am

libvlc_LTLIBRARIES += libgain_plugin.la
SOURCES_gain = gain.c
```

Bootstrap the Build System: From the root of the VLC source directory, run the bootstrap script to make the build system aware of your new files.

```
./bootstrap
```

2. **Configure and Compile**:

```
./configure
make
```

This will compile all of VLC, including your new plugin. The resulting shared library (`libgain_plugin.so`) will be in `vlc/modules/audio_filter/`.

3. **Install or Run**: You can either run sudo make install to install it system-wide, or you can run VLC directly from the build directory, which is safer for testing:

```
./vlc
# if you want to run Qt explicitly
vlc --intf=qt
```

E. Activating Your New Filter

You can now enable your "Gain" filter just like any other:

- **GUI**: `Go to Tools -> Preferences -> All -> Audio -> Filters` and check the "Gain" box.
- **Command Line**:

```
./vlc my_song.mp3 --audio-filter=gain
```

Your custom audio logic will now be applied to any media you play!