УДК 004

Обработка аудио файла WAV на примере изменения громкости звука

Вихляев Дмитрий Романович
Приамурский государственный университет имени Шолом-Алейхема
Студент

Аннотация
Данная статья содержит описание программы, которая способна изменять громкость звука аудио файла формата «wav». Программа написана на языке программирования C, с использованием только методов стандартной библиотеки. Результатом исследования станет готовая программа с подробным описанием её реализации.
Ключевые слова: C, wav, обработка звука.

Processing an audio WAV file using the example of changing the volume of sound

Vikhlyaev Dmitry Romanovich
Sholom-Aleichem Priamursky State University
Student

Abstract
This article contains a description of a program that is able to change the sound volume of an audio file in the wav format. The program is written in the C programming language, using only the methods of the standard library. The result of the study will be a ready-made program with a detailed description of its implementation.
Keywords: C, wav, sound processing.

1 Введение

1.1 Актуальность

WAV является короткой формой Wave Audio File Format (реже именуемой как Аудио для Windows). Этот формат используется для хранения несжатого звука в исходном качестве записи. Стандарт WAV совместим с операционными системами Windows, Macintosh и Linux. Формат учитывает отличия процессоров Intel, например, расположение порядка младшего байта к старшему байту. Является одним из самых популярных форматов для переноса звуковой информации и с каждым годом люди находят к нему новые применения. Однако не всегда он соответствует ожиданиям. Иногда звуковой файл имеет низкие значения громкости, которые доставляют большой дискомфорт, даже при усилении мощности колонок. Тот, кто передаёт звук, может говорить слишком тихо или невнятно, в таком случае человек может не услышать всю предоставляемую информацию. В другом

случае сам звуковой файл может иметь слабый звук. В таких случаях возникает необходимость использовать сторонние программы для усиления громкости звука. На помощь приходят современные технологии обработки звука, с помощью которых каждый может настроить звуковую передачу информации под себя.

1.2Обзор исследований

М.Б.Столбов и С.В.Перелыгин описали алгоритм цифровой обработки сигналов двух микрофонов для улучшения качества звука [1]. А.В.Чапурин продемонстрировал нововведения в обработке звука [2]. Р.А.Свентковский реализовал распознавание звуков речи основе методов обработки спектра с использованием рекурсивных цифровых фильтров [3]. Е.Г.Алексеева провёл анализ алгоритмов статистической обработки данных для моделирования локализации источников звука [4]. Н.В.Савченко привёл пример использование архитектуры CUDA для решения задач обработки и анализа звука [5].

1.3Цель исследования

Цель исследования – используя язык программирования С реализовать программу способную, изменять громкость звука «wav» файла. На входе программа принимает аудио файл, на выходе создаёт изменённую копию входного файла.

2 Материалы и методы

Для создания программы используется звуковой файл формата «wav», и язык программирования С.

3 Результаты и обсуждения

Использование формата WAV является общепринятым, благодаря его простой структуре, которая в большой степени основана на формате файлов RIFF. RIFF формат играет роль обертки для различных кодеков аудио сжатия. Благодаря чему, формат WAV не испытывает никаких примесей для различного программного обеспечения или аппаратных плееров, он поддерживается практически везде.

PCM — импульсно-ко́довая модуля́ция, используется для оцифровки аналоговых сигналов. При импульсно-кодовой модуляции аналоговый передаваемый сигнал преобразуется в цифровую форму посредством трёх операций: дискретизации по времени, квантования по амплитуде и кодирования.

В WAV используется стандартная RIFF структура. Данные можно условно разделить на 3 части заголовок, секция формата, данные. Для хранения обычного несжатого звука, как правило, применяется кодирование методом линейной импульсно-кодовой модуляции.

Каждый файл WAV имеет вначале заголовочные данные хранящие информацию о содержании. Стандартный заголовок имеет длину в 44 байта.

Если извлечь четыре байта подряд, начиная с восьмого можно определить, что данный файл является именно WAV файлом.

Два байта, начиная с двадцать второго, указывают на количество каналов, что позволяет определить количество байт на канал (моно 1 байт, стерео 2 байта) (табл. 1).

Таблица 1. Заголовок WAV файла

Размер поля в байтах	Первый байт поля	Название поля	Описание поля
4	0	chunkId	Содержит символы «RIFF» в ASCII кодировке 0x52494646. Является началом RIFF-цепочки.
4	4	chunkSize	Оставшийся размер цепочки, начиная со следующего поля (format).
4	8	format	Содержит символы «WAVE» 0x57415645
4	12	subchunk1Id	Начало секции формата данных. Содержит символы "fmt " 0x666d7420
4	16	subchunk1Size	Размер секции, начиная с этой позиции (16 для формата PCM).
2	20	audioFormat	Аудио формат сжатия. Для PCM = 1 (Линейное квантование).
2	22	numChannels	Количество каналов. Моно = 1, Стерео = 2
4	24	sampleRate	Частота дискретизации. 8000 Гц, 44100 Гц и т.д.
4	28	byteRate	Количество байт, переданных за секунду воспроизведения.
2	32	blockAlign	Количество байт для одного сэмпла, включая все каналы.
2	34	bitsPerSample	Количество бит в сэмпле (глубина, точность звучания). 8 бит, 16 бит и т.д.
4	36	subchunk2Id	Содержит символы «data» 0x64617461
4	40	subchunk2Size	Количество байт в области данных.
_	44	data	Непосредственно WAV-данные.

В некоторых случаях размер заголовка может быть не определён. Например, после конвертации формата MP3 в WAV при использовании технологии FFMPEG. Тогда количество подцепочек может быть больше, чем две, помимо рассмотренных подцепочек fmt и data будет содержаться ещё одна LIST перед областью данных. В таком случае необходимо пропустить ненужные подцепочки, пока не встретится data, читая ID подцепочки, и пропускать данные, основываясь на её размере.

Первым делом для работы необходимо открыть исходный файл на чтение в бинарном режиме. Узнать полный размер файла можно с помощью данных заголовка или встроенных функций fseek и ftell.

Заголовок можно считать, сохранив в массив без знаковых восьми битных целых чисел.

Поле данных, при работе со стереозвуком, удобно расположить в массив без знаковых шестнадцатеричных целых чисел.

Так как данные в заголовке должны остаться неизменными, первый массив будет перенесён в новый файл как есть, а второй с данными будет преобразован и склеен с первым (рис. 1).

```
#define HEADER_SIZE 44

int main(int argc, char* argv[])
{
    FILE* input = fopen(argv[1], "rb");
    if (input == NULL)
    {
        printf("Could not open file.\n");
        return 1;
    }
    //перевести курсор на конец файла
    fseek(input, 0, SEEK_END);
    //получение номера байта указанного курсором
    int size_file = ftell(input);
    //перевести курсор на начало файла
    fseek(input, 0, SEEK_SET);

    int slamp_count=(size_file-HEADER_SIZE)/sizeof(uint16_t);
    uint16_t *slamp = (uint16_t*)malloc(sizeof(uint16_t)*slamp_count);
    if (slamp == NULL)
    {
        return 3;
    }
    uint8_t header[HEADER_SIZE];
    fread(header,sizeof(uint8_t),HEADER_SIZE,input);

    fread(slamp,sizeof(uint16_t),slamp_count,input);

    fclose(input);
```

Рис. 1. Извлечение данных из файла

Каждый элемент массива данных хранит значения сэмпла. Семпл — N-байтовый элемент массива, предварительно дискретизированный в цифровой формат аналоговый сигнал. Увеличение значения каждого сэмла приведет к

усилению громкости звука. Таким образом, получается, что если умножить каждый сэмпл на 2, то громкость звука увеличится в 2 раза, или уменьшению, если умножить на число меньше единицы.

Стоит помнить, что усиление громкости ограничено количеством возможных значений 2^(8*k), где k количество каналов в файле. Если при умножении произойдёт переполнение, то итоговый файл будет иметь помехи при звучании (рис.2).

```
float format = argv[3];
for(int i=0;i<slamp_count;i++)
{
    slamp[i]*=format;
}

FILE* output = fopen(argv[2], "wb");
if (output == NULL)
{
    printf("Could not open file.\n");
    return 2;
}

fwrite(header,sizeof(uint8_t),HEADER_SIZE,output);

fwrite(slamp,sizeof(uint16_t),slamp_count,output);
fclose(output);
```

Рис. 2. Запись изменённых данных в новый файл

Ниже представлены графики зависимости амплитуды сигнала от времени над одним и тем же звуковым файлом с разным значением сэмплов (рис. 3,4,5).

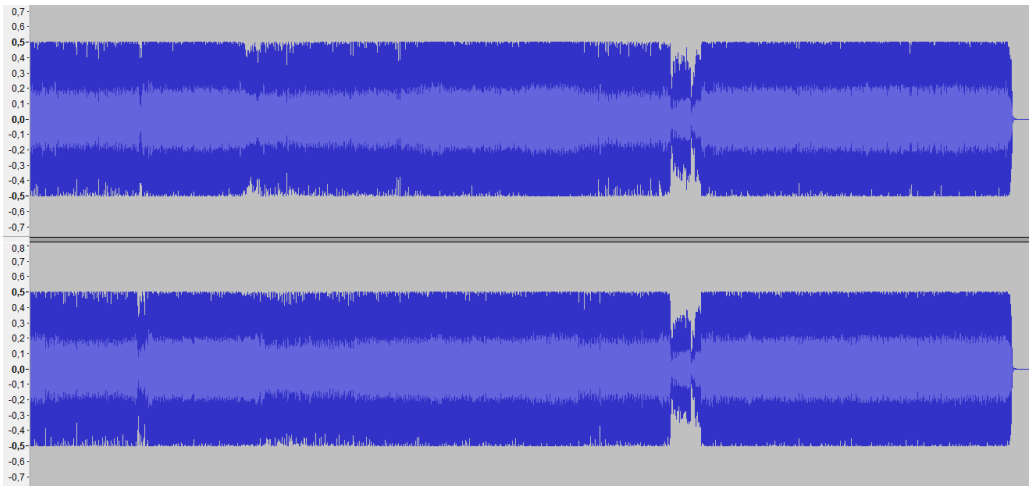

Рис. 3. График исходного аудио файла

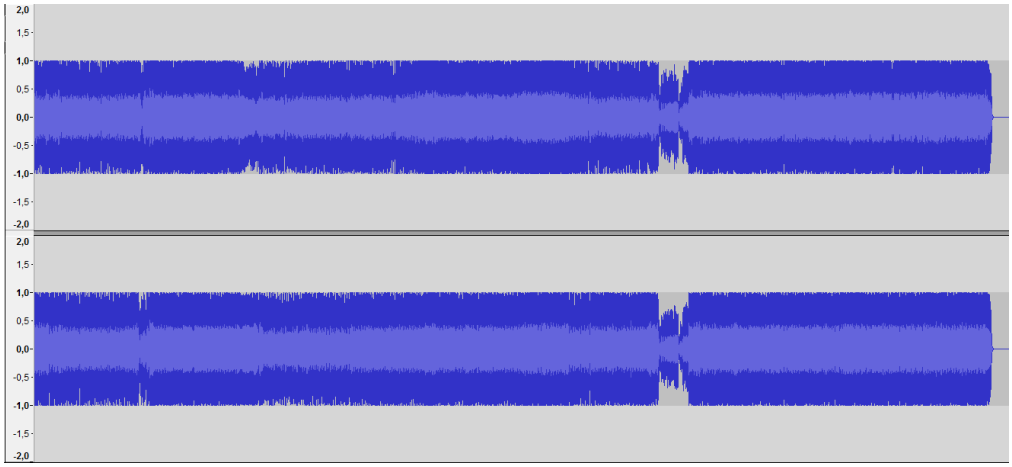

Рис. 4. График аудио файла с увеличенным значением сэмплов в 2 раза

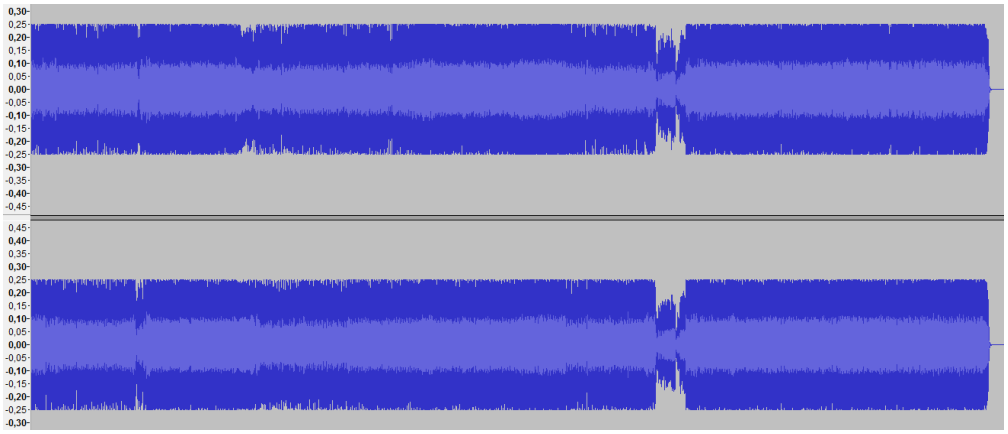

Рис. 5. График аудио файла с увеличенным значением сэмплов в 0.5 раза

Таким образом, было описано, как можно на низкоуровневом доступе обрабатывать WAV файл, и представлено описание программы на языке программирования C для изменения громкости на примере стереозвука.

Библиографический список

1. Столбов М.Б., Перелыгин С.В. Алгоритм цифровой обработки сигналов двух микрофонов для улучшения качества звука// В сборнике: Актуальные проблемы радио- и кинотехнологий. материалы II Международной научно-технической конференции. 2018. С. 121-129.
2. Чапурин А.В. Обработка звука как возможность услышать что-то новое// Научный альманах. 2016. № 2-2 (16). С. 420-425.
3. Свентковский Р.А. Распознавание звуков речи и идентификация диктора на основе методов обработки спектра с использованием рекурсивных цифровых фильтров// Информатизация и связь. 2013. № 3. С. 99-103.
4. Алексеева Е.Г. Анализ алгоритмов статистической обработки данных для моделирования локализации источников звука // Вопросы радиоэлектроники. 2010. Т. 3. № 2. С. 44-49.
5. Савченко Н.В. Использование архитектуры CUDA для решения задач

обработки и анализа звука // Вестник научных конференций. 2017. № 1-5 (17). С. 151-152.
6. Маринчук А.С. Создание аудио проигрывателя с помощью microsoft visual studio и библиотеки naudio// Постулат. 2020. № 1 (51). С. 132.
7. Семченко Р.В., Еровлев П.А. Создание аудио плеера на андроид смартфон// Постулат. 2019. № 12 (50). С. 14.
8. Кирьянова Л.Г. Цифровая обработка аудио- и видеоинформации // Информационные технологии. Радиоэлектроника. Телекоммуникации. 2012. № 2-2. С. 225-231.
9. Киселева С.Д., Недильченко О.С., Линев Ф.А. Некоторые методы анализа музыкальных фрагментов // Молодежный научно-технический вестник. 2017. № 3. С. 19.
10. Добровольский Р.В., Ким Т.А., Сотников А.А. Цифровая фильтрация аудио сигналов в режиме реального времени на базе процессора tms320c5515 // Технологии инженерных и информационных систем. 2019. № 1. С. 11-20.
11. Егоров Д.П., Кутуза Б.Г. Распознавание нотной партитуры по цифровому аудио сигналу //Актуальные проблемы современного образования. 2018. Т. 2. С. 70-73.
12. Вихляев Д.Р. Регулирование скорости звука с помощью языка программирования python // Постулат. 2021. № 10 (72).