DOI 10.37882/2223–2966.2021.06.35

ЦИФРОВАЯ БИБЛИОТЕКА ЧТЕНИЯ И ЗАПИСИ АУДИОДАННЫХ LIBSNDFILE: ТЕХНИЧЕСКАЯ СТРУКТУРА, ВОЗМОЖНОСТИ ИСПОЛЬЗОВАНИЯ И ПЕРСПЕКТИВЫ РАЗВИТИЯ

DIGITAL LIBRARY FOR READING AND WRITING LIBSNDFILE AUDIO DATA: ENGINEERING STRUCTURE, APPLICATIONS AND FUTURE EVOLUTION

V. Taran
R. Gilyarevski

Summary. The article touches upon the aspects of digital reading and audio data writings and forms the reader's imagining about open libsndfile library. Own experience and manuals in English provide the authors an opportunity to investigate key library service. The fundamental libsndfile library technical potential and key opportunities for coding and decoding of audio material are disclosed. Basic commands set by the operator contributing qualitative and effective objective fulfillment are reviewed, arranged and translated to Russian. The examples of the wide–spread operations requiring the operator timely input of the commands for on-line problem solutions related to audio material coding and decoding are demonstrated. Extended interpretations of popular commands qualified for audio processing procedures are presented.

Keywords: libsndfile, digital library, audio data, integer-valued data, reading and writing of audio material.

Таран Василий Васильевич
К.культурологии, ФГБУН «Всероссийский институт научной и технической информации РАН»
allscience@lenta.ru
Гиляревский Руджеро Сергеевич
Д.филол.н., профессор, заслуженный деятель науки РФ, МГУ им. М.В. Ломоносова; ФГБУН «Всероссийский институт научной и технической информации РАН»
ruggero29@gmail.com

Аннотация. Статья затрагивает аспекты цифрового чтения и записи аудиоданных и формирует представление читателя об открытой библиотеке libsndfile. Авторы статьи на основе собственного опыта и англоязычных инструкций анализируют ключевой функционал рассматриваемой библиотеки. В статье раскрывается базовый технический потенциал библиотеки и её основные возможности по кодированию и декодированию звукового материала. В обзорном виде систематизированы, переведены на русский язык и изложены основные команды, способствующие качественному и эффективному достижению целей, поставленных оператором технологического процесса. В статье приведены примеры наиболее популярных операций, требующих от потенциального оператора своевременного ввода команд для оперативного решения задач, связанных с кодированием и декодированием аудиоматериала, а также представлены расширительные трактовки популярных команд, характерные процедурам обработки аудиоматериала.

Ключевые слова: libsndfile, цифровая библиотека, аудиоданные, целочисленные данные, чтение и запись аудиоматериала.

Компьютерная обработка аудиосигнала, фундаментально базирующаяся на передовых алгоритмах просчёта спектральных блоков, формирующих основной состав аудиоформы, требует особого хранения и упорядочивания аудиоданных с целью их записи, перезаписи, и последующего воспроизведения на различных программно-аппаратных платформах. Специально для такого рода действий предусмотрены особые цифровые библиотеки, позволяющие управлять процессами обращения к аудиоданным.

По сути, такие библиотеки являются посредником между программой обработки аудиоматериала и обслуживающей её операционной средой. Объектом содержания нашего мини-исследования, под названием Libsndfile[1], как раз и является разработка подобного рода.

Строго говоря libsndfile — это библиотека-посредник, написанная на языке программирования Си (далее по тексту — С), осуществляющая чтение и запись аудиосигнала посредством единого командного интерфей-

[1] Прим. автора. (Таран В.В.). Далее по тексту наименование библиотеки пишется с маленькой (сторочной) буквы как в англоязычных источниках в целях сохранения традиционного представления аббревиатуры согласно программному коду. Исключения составляют только начало предложений, в соответствии с правилами русского языка.

Рис. 1. Логотип библиотеки libsndfile*. Графическое воспроизведение логотипа и его повторная зарисовка — Таран Василий Васильевич, все права на оригинальное исполнение логотипа принадлежат его законным владельцам.

* Прим. автора. (Таран В.В.). Все права на использование логотипа принадлежат его законным владельцам. Логотип воспроизводится в статье исключительно в научных целях для визуальной идентификации данного программного продукта.

са[1]. При бурном развитии компьютерных наук и технологий, ориентированных на персональную обработку аудиоматериала, рынок информационных технологий изобилует продуктами подобного рода, однако не все из них отвечают следующим критериям:

- наличие собственной микросреды исполнения команд;
- обладание возможностями качественной интеграции в приложения обработки аудиоданных;
- распространение по лицензии GNU Lesser General Public License (LGPL);
- поддерживание кроссплатформенности при исполнении программного кода;
- производство произвольной нормализации при чтении данных с плавающей запятой из файлов, содержащих целочисленные данные;
- осуществление записи заголовка файла, не закрывая файл;
- наличие возможности выполнения запроса в библиотеку, описывающую все поддерживаемые форматы и получение данных, описывающих каждый формат по отдельности в строчном виде.

Библиотека libsndfile соответствует каждому из приведённых критериев. Но, к сожалению, проведённый авторами анализ текущей русскоязычной научно-технической литературы, предваряющий ход данного исследования по рассматриваемому нами направлению, не выявил существенных положений, способных охарактеризовать достаточно полно картину особенностей данной библиотеки.

В этой связи авторы сочли возможным предпринять попытку расширения представления об этом феномене, взяв за основу технические инструкции, опубликованные на официальном сайте библиотеки libsndfile, а также руководствуясь некоторыми другими англоязычными источниками, затрагивающими тематику данной статьи.

Таким образом, информационная база нашего мини — исследования зиждется исключительно на электронных источниках: на технических инструкциях, правилах и регламентах, затрагивающих функционирование библиотеки libsndfile [1,2,3,4,5,6,7].

Официальное распространение данная библиотека получила в 1999 году, после презентации её технического пресс-релиза. Версия получила цифровую идентификацию 0.0.8., хотя она тестировалась на разных платформах — Microsoft Windows 3.1 в 1990 году, под Linux в 1995 году, т.е. задолго до представления официальной версии библиотеки.

В результате многочисленные тесты, портирование и отладка кода привели к официальной версии. Автором разработки является Эрик де Кастро Лопо[2] (Erik de

[1] Прим. автора. (Таран В.В.). Интерфейс библиотеки подразумевает набор приглашений командной строки, который предназначен для оперативного управления кодированием и декодированием, а также распределением и перераспределением цифровых аудиоданных. Каждое из приложений связано между собой и может выполнять команды одновременно, в режиме реального времени. К примеру, есть команды устанавливающие мосты между приглашениями для отработки конкретных операций, например расчёт пиковых значений для каждого канала или расчёт нормированного значения для каждого канала. В первом случае это команда SFC_CALC_MAX_ALL_CHANNELS в одном приглашении, и во втором случае команда SFC_CALC_NORM_MAX_ALL_CHANNELS в параллельном приглашении. Оператор * — позволит вывести статистические данные в обоих приглашениях.

[2] Прим. автора. (Таран В.В.). Эрик де Кастро Лопо (англ. Erik de Castro Lopo) — практикующий профессиональный программист широкого профиля, специализирующийся преимущественно на языке Haskell. В число его основных научных интересов входит: компьютерные вычисления, функциональное программирование, численные методы, криптовалюты, язык программирования Haskell, цифровая обработка сигналов. Владеет практиками программирования приложений под операционные системы MS Windows и Linux. Является основным автором двух специализированных библиотек чтения и записи аудиоданных написанных на языке C (libsndfile и libsamplerate). Владеет техниками программирования на OCaml (**O**bjective **C**aml) с 2004 года и Haskell — с 2006 года. Имеет второй по величине вклад создание компилятора DDS (компилятор для языка, подобного Haskell, написанного на языке Haskell). Входит в группу сопровождения Debian Haskell и группу технического мониторинга пакетов Haskell на Github.

Castro Lopo). Суть библиотеки заключается в возможности преобразования массивов данных в числа с плавающей запятой для приложений обработки звука. Данный аспект во многом упрощает работу программиста при отлаживании систем ввода-вывода данных на программном уровне и позволяет обходиться без упорядочивания байтов. Рассматриваемая нами библиотека нашла своё применение в области коммерческого — Adobe Audition[1] и некоммерческого — Audacity*[2] программного обеспечения, нацеленного на качественную обработку звука, что также подтверждает её востребованность и перспективы развития на мультимедийном рынке. Исходный программный код библиотеки libsndfile регулируется лицензией GNU[3] (Стандартная общественная лицензия ограниченного применения) и распространяется согласно редакциям 2.1 и 3.0[4]. Libsndfile первоначально была спроектирована для запуска и компиляции в операционной среде Linux. Она также хорошо зарекомендовала себя практически на всех Unix-подобных операционных системах, включая коммерческую ветку Unix — MacOS X. Библиотека представлена в двух форматах: 32-х разрядная версия и 64-х битная версия. Технически библиотека сформирована как для чтения аудиоданных в порядке прямого следования байтов — WAVE, так и для обратного порядка байтов — AIFF и компиляции, а также для корректной работы на процессорах обработки прямого порядка байтов (Intel, DEC/Compaq Alpha) и процессорах обратного порядка байтов (Motorola 68k, Power PC, MIPS, Sparc).

Начиная с версии 1.0.18, библиотека осуществляет поддержку чтения и записи FLAC и Ogg/Vorbis. Проектные характеристики библиотеки позволяют настроить её функционал в соответствии с техническими параметрами, заданными программистом, что в перспективе позволит расширить её базу чтения и записи новых стандартов и аудиоформатов. В операционной среде MS Windows библиотека станет хорошим помощником в организации мостов импорта и экспорта, битовых аудиомассивов между различными программами, направленными на редактирование и улучшение качества аудиоматериала. Библиотека libsndfile имеет собственную линейку командных файлов (табл. 8), которые позволяют эффективно управлять процессами чтения и записи аудиоданных через приглашение командной строки.

При определённой компиляции кода библиотеку также можно использовать как локальный программный трансмиттер для перевода прямых[5] аудиопотоков из одной программы звукообработки в другую. Почему это важно? Ответ на этот вопрос лежит в многолетних практиках компьютерной аудиоинженерии. Очень часто программистам, звукоинженерам, звукооператорам и техническим экспериментаторам в области звука требуются условия, при которых необходимое количество аудиопотока должно распределяться между программами обработки звука, а в некоторых особых случаях, перераспределяться и перенаправляться далее на внешние виртуальные (программная маршрутизация) или физические (вывод на колонки) источники воспроизведения звука. Libsndfile имеет все возможности для реализации описанных выше процедур, благодаря

Эрик де Кастро Лопо получил степень бакалавра в области компьютерных технологий в Университете Ньюкасла в Австралии. После окончания университета Эрик работал инженером-проектировщиком аппаратного обеспечения, прежде чем перейти на низкоуровневое системное программирование на языке C. Он заново открыл для себя функциональное программирование в 2004 году, изучив OCaml, а затем четыре года спустя перешёл на язык Haskell. Он присоединился к IOHK* в 2018 году.

* **IOHK** (Input-Output Hog Kong) — это инжиниринговая компания, которая разрабатывает блокчейн-решения для академических учреждений, корпораций и государственных организаций. Компанию возглавляет Чарльз Хоскинсон (Charles Hoskinson) — бывший генеральный директор команды Ethereum и Джереми Вуд (Jeremy Wood) — исполнительный помощник команды Ethereum.

[1] Прим. автора. (Таран В.В.). Подробнее об использовании библиотеки libsndfile в Adobe Audition вы можете ознакомиться в Уведомлениях о программном обеспечении третьих лиц и/или Дополнительных условиях и положениях, опубликованных по адресу*: https://www.adobe.com/content/dam/cc/en/products/eula/third_party/pdfs/CS6_readme_05072012.pdf (дата обращения к электронному источнику: 23.04.2021).

* Документ представлен в формате PDF, об использовании библиотеки libsndfile в Adobe Audition смотрите страницу 149.

[2] Audacity* — официальный сайт компьютерной программы редактирования аудиоматериала: www.audacityteam.org. О поддержке libsndfile в Audacity* на Unix-подобных системах можно прочитать в предписании по адресу: http://manpages.ubuntu.com/manpages/trusty/man1/audacity.1.html. Об ошибках экспорта WAVE-файлов в Audacity* сказано здесь: https://forum.audacityteam.org/viewtopic.php?t=57218.

[3] Прим. автора. (Таран В.В.). GNU Lesser General Public License (LGPL)-это лицензия на свободное программное обеспечение, опубликованная Фондом свободного программного обеспечения (FSF)*. *FSF — это некоммерческая организация, основанная Ричардом Столлманом (Richard Stallman) 4 октября 1985 года для поддержки движения за свободное программное обеспечение, которое продвигает всеобщую свободу изучения, распространения, создания и модификации компьютерного программного обеспечения с предпочтением распространять программное обеспечение на условиях сохранения прав копирования («share alike» — участие в равных долях). Например, с собственной лицензией GNU General Public License. FSF зарегистрирована в Бостоне, штат Массачусетс, США, где она также базируется.

[4] Прим. автора. (Таран В.В.). Автором данной библиотеки специально предусмотрено использование программного кода с лицензиями в двух редакциях. Объясняется это в первую очередь гибкостью интеграции данной программной разработки, чтобы авторы других коммерческих компьютерных программ и свободные разработчики смогли более эффективно использовать код библиотеки как интегрированного в локальную программную среду продукта. Разработчик вправе сам выбирать какую из редакций лицензии ему удобнее использовать. Libsndfile является динамически загружаемой библиотекой и отлично совмещается со свободно-распростроняемым программным обеспечением имеющим открытый исходный код, программным обеспечением с определёнными условиями распространения а также с уловно-бесплатными программами и программами использующими оригинальный код закрытого типа.

[5] Прим. автора. (Таран В.В.). Имеются ввиду линейные аудиопотоки с возможностью их чтения и записи внутри выбранной программы.

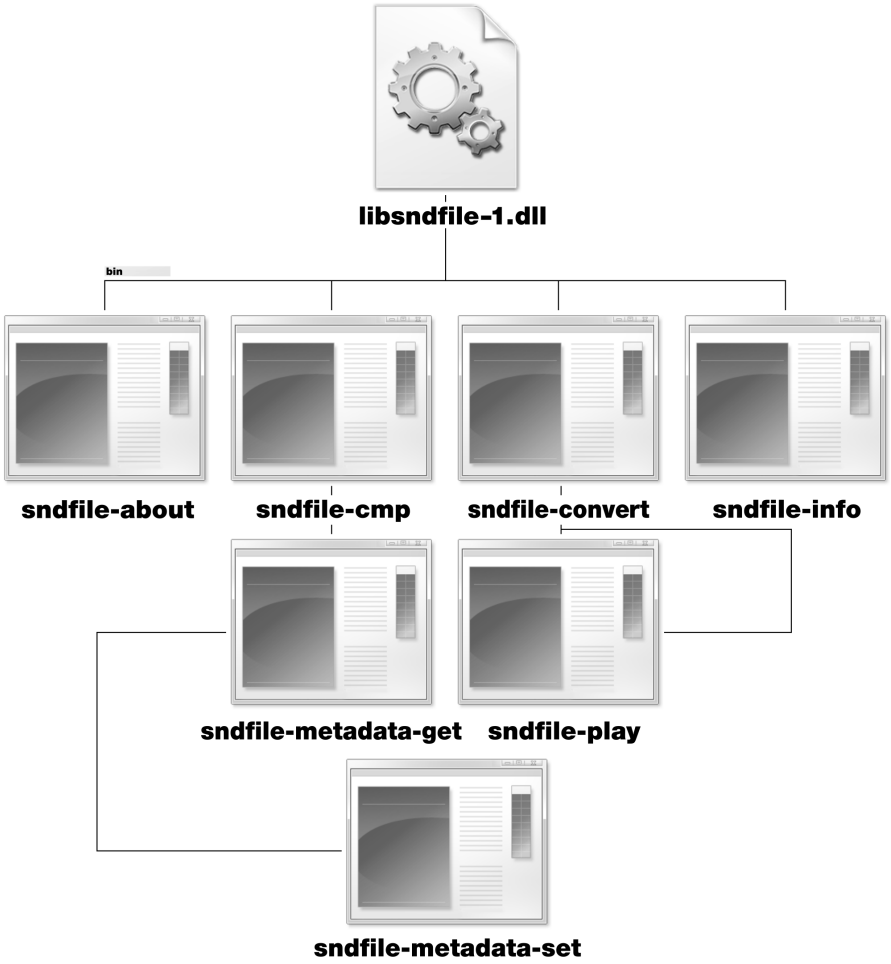

Рис. 2. Набор командных файлов, реализованный в виде приглашений командной строки MS Windows.

открытости кода и его развитой технической структуре. При желании реализации подобных возможностей можно обратиться к функции открытия закодированного аудиофайла библиотеки (по указанному пути) в операционной системе MS Windows — обращение будет следующим (рис. 2):

specific sf_wchar_open() — берёт имя файла, которое было закодировано UTF16_BE (specific sf_wchar_open() → UTF16_BE). Библиотека libsndfile имеет расширенный арсенал функций, способных удовлетворить многоаспектные запросы специалистов в области изучения и компьютерного редактирования аудиоматериала, расшифровка которых непременно получит дальнейшее отражение в статьях, публикуемых авторами данного исследования. Проводники командных файлов (в виде приглашений) в MS Windows располагаются в директории \bin. Они имеют перекрёстную структуру и различные сценарные зависимости. Поэтому их следует рассматривать комплексно как единый механизм, зависящий от dll-файла библиотеки. Структура сценарных зависимостей может быть представлена следующим образом:

1) libsndfile-1.dll — программный файл библиотеки libsndfile, хранящийся в виде цифровой библиотеки.

2) sndfile-about — информация о текущем состоянии библиотеки. Предоставляет технические характеристики для отладки кодирования и декодирования потенциального аудиоматериала.

3) sndfile-cmp — производит сравнение двух аудиофайлов для выявления схожих характеристик, включая количество каналов, частоту дискретизации, хронометраж аудиофайла, т.е. те данные, которые должны совпадать. Другие различия: метаданные строки, в том числе название композиции, исполнитель, автор и т.д., игнорируются.

4) sndfile-convert — преобразует аудиоданные из одного установленного формата в другой.

Таблица 1. Поддерживаемые форматы преобразования аудиоданных при использовании команды sndfile-convert.

ФОРМАТЫ ПРЕОБРАЗОВАНИЯ АУДИОДАННЫХ ПО КОМАНДЕ SNDFILE-CONVERT		
№	**Расширение**	**Расширенное название**
1	aif	AIF(Apple/SGI)
2	au	AU Sun/NeXT
3	wav	WAV(WAVE) / Microsoft
4	snd	AU (Sun/NeXT)
5	raw	RAW (header-less)
6	gsm	RAW (header-less)
7	vox	RAW (header-less)
8	paf	PAF (Ensoniq PARIS, big-endian)
9	fap	PAF (Ensoniq PARIS, little-endian)
10	svx	IFF (Amiga IFF/SVX8/SV16)
11	nist	SPHERE (NIST SPeech HEader Resources)
12	sph	SPHERE (NIST SPeech HEader Resources)
13	voc	VOC (Creative Labs)
14	ircam	SF (Berkeley/IRCAM/CARL)
15	sf	SF (Berkeley/IRCAM/CARL)
16	w64	W64 (SoundFoundry WAVE64)
17	mat	MAT4 (GNU Octave 2.0 / Matlab 4.2)
18	mat4	MAT4 (GNU Octave 2.0 / Matlab 4.2)
19	mat5	MAT5 (GNU Octave 2.1 / Matlab 5.0)
20	pvf	PVF (Portable Voice Format)
21	xi	XI (FastTracker 2)
22	htk	HTK (HMM Tool Kit)
23	sds	SDS (Midi Sample Dump Standard)
24	avr	AVR (Audio Visual Research)
25	wavex	WAVEX (MS WAVE with WAVEFORMATEX)
26	sd2	SD2 (Sound Designer II)
27	flac	FLAC (FLAC Lossless Audio Codec)
28	caf	CAF (Apple Core Audio File)
29	wve	WVE (Psion Series 3)
30	prc	WVE (Psion Series 3)
31	ogg	OGG (OGG Container format)
32	oga	OGG (OGG Container format)
33	mpc	MPC (Akai MPC2k)
34	rf64	RF64 (RIFF 64)

Таблица 2. Вспомогательные инструменты команды sndfile-convert.

№	**Инструмент**	**Оператор***	**Поддерживаемая операция преобразования**
1	override-sample-rate=rate	[=]	Делает так, чтобы на входе использовалась частота дискретизации частоты в Гц.
2	endian=little	[=]	Делает так, чтобы выходной файл использовал мало конечных данных.
3	endian=big	[=]	Делает так, чтобы выходной файл использовал данные «big endian».
4	endian=cpu	[=]	Делает так, чтобы выходной файл использовал порядок следования байтов CPU.
	normalize	[=]	Нормализует аудиоданные в выходном файле.

* Прим. автора. (Таран В.В.).Оператор [=] устанавливает значения расширений для перечисленных инструментов в диапазоне [rate], [little], [big], [cpu], [normalize]. В последнем случае [normalize]= [normalize] используется как универсальная опция в операции нормализации аудиоданных в выходном файле, т.е. может работать во взаимодействии с первыми четырьмя расширениями.

Таблица 3. Кодировки, поддерживаемые при преобразовании форматов командой sndfile-convert.

№	Кодировка	Расшифровка кодировки
1	pcms8	signed 8 bit pcm
2	pcmu8	unsigned 8 bit pcm
3	pcm16	16 bit pcm
4	pcm24	24 bit pcm
5	pcm32	32 bit pcm
6	float32	32 bit floating point
7	ulaw	ULAW
8	alaw	ALAW
9	ima-adpcm	IMA ADPCM (WAV only)
10	ms-adpcm	MS ADPCM (WAV only)
11	gsm610	GSM6.10 (WAV only)
12	dwvw12	12 bit DWVW (AIFF only)
13	dwvw16	16 bit DWVW (AIFF only)
14	dwvw24	24 bit DWVW (AIFF only)
15	vorbis	Vorbis (OGG only)

Таблица 4. Список подкоманд, выводящих данные на печать.

ПАРАМЕТРЫ ВЫВОДА ДАННЫХ НА ПЕЧАТЬ		
№	Подкаманды вывода данных на печать	Описание подкаманд вывода данных на печать
1	all	Все метаданные
2	bext-description	Описание
3	bext-originator	Информация о составителе
4	bext-orig-ref	Ссылка на составителя
5	bext-umid	Уникальный идентификатор аудиовизуального материала, UMID (**U**nique **M**aterial **Id**entifier)
6	bext-orig-date	Дата компиляции
7	bext-orig-time	Время компиляции
8	bext-coding-hist	История кодирования
9	str-title	Название (включая заголовок титра)
10	str-copyright	Авторское право
11	str-artist	Исполнитель
12	str-comment	Комментарий
13	str-date	Дата создания
14	str-album	Альбом
15	str-license	Лицензия

Официально поддерживаются следующие форматы файлов (таблица 1).

sndfile-convert имеет также ряд вспомогательных инструментов, направленных на адаптивное преобразование аудиосигнала. Среди них можно отметить следующие (таблица 2).

Произвольный параметр преобразования дает возможность задать кодировку данных для выходного файла, в настоящее время поддерживаются следующее типы кодировок (таблица 3).

В том случае, если для выходного файла не указана кодировка, sndfile-convert попытается использовать кодировку входного файла. Это не всегда будет работать, поскольку большинство форматов контейнеров (например, WAV/WAVE, AIFF и т.д.) поддерживают только небольшое подмножество кодировок (например, 16-битный PCM, a-law, Vorbis и т.д.).

Таблица 5. Параметры аргументов, указывающих метаданные при эксплуатации команды sndfile-metadata-set.

АРГУМЕНТЫ ДЛЯ ОПИСАНИЯ ВИДА МЕТАДАННЫХ		
№	**Аргумент**	**Описание аргумента**
1	bext-description	Описание
2	bext-originator	Составитель
3	bext-orig-ref	Ссылка на составителя
4	bext-umid	Уникальный идентификатор аудиовизуального материала, UMID (Unique Material Identifier)
5	bext-orig-date	Дата компиляции
6	bext-orig-time	Время компиляции
7	bext-coding-hist	История кодирования
8	bext-time-raf	Отсчёт времени
9	str-comment	Комментарий
10	str-title	Название
11	str-copyright	Авторское право
12	str-artist	Исполнитель
13	str-date	Дата
14	str-album	Альбом
15	str-license	Лицензия

5) sndfile-info — командный файл целью которого служит отображение основной информации об аудиоматериале[1].

6) sndfile-metadata-get — отображает метаданные bext и строки, хранящиеся в аудиоматериале. Следующие параметры указывают, что печатать (таблица 4).

7) sndfile-metadata-set[2] — устанавливает метаданные «bext»[3] и строки в аудиофайле, если формат их поддерживает. Если файл не содержит фрагмента BEXT, который должен быть улучшен, необходимо использовать второй конспект (в котором создается другой выходной файл, способный хранить метаданные). Этот файл перезаписывается, если он уже существует. Следующие параметры принимают аргумент, указывающий метаданные (таблица 5).

Следующие параметры не принимают аргументов (таблица 6).

8) sndfile-play — воспроизводит один или несколько звуковых файлов в различных операционных системах, используя стандартные аудиовыходы APIs[4]. В следующей таблице приведены сводные данные о том, какой аудио API используется в следующих разновидностях операционных систем (таблица 7).

[1] Прим. автора. (Таран В.В.). Может отображать формат количество каналов частоту дискретизации и хронометраж аудиоматериала. Может распознаваться тип трансляции и отображение самой транслируемой информации (BWF). Также отображает фрагмент корзины WAVE-файла (или связанного с ним файла). Поддерживает отображение карты канала включая отображение информации об инструменте: базовой ноте усилении скорости и ключевые циклические точки.

[2] Прим. автора. (Таран В.В.). Подробнее о функционале команд устанавливающих значения метаданных на операционных системах Linux (различных его дистрибутивах), а также о прочей технической информации можно прочитать по следующим адресам:
- https://man.archlinux.org/listing/extra/libsndfile/
- https://github.com/libsndfile/libsndfile/tree/master/programs

Дата обращения к электронным источникам: 23.04.2021.

[3] Прим. автора. (Таран В.В.). Bext — формат файла, предназначенный для обмена аудиоматериалами между различными средами вещания и оборудованием, основанным на различных компьютерных платформах. Основываясь на формате аудиофайла Microsoft WAVE, Broadcast Wave (Broadcast WAVE Audio File Format) добавляет необходимый фрагмент «**B**roadcast Audio **Ext**ension» (bext) для хранения минимальной информации, которая считается необходимой для приложений широковещательной передачи. При обращении к фрагменту данных в качестве заголовка записывается как «BEXT» — широковещательное аудиорасширение.

[4] Прим. автора. (Таран В.В.). API [**A**pplication **P**rogramming **I**nterface, во множ. числе — API(s)] интерфейсы прикладного программирования, позволяющие адаптировать различные технико-программные средства под конкретные нужды оператора.

Таблица 6. Параметры, при которых аргументы не принимаются в период указания метаданных при работе команды sndfile-metadata-set.

№	bext / sndfile-metadata-set	Описание параметров
1	bext-auto-time-date	Устанавливает время и дату BEXT на текущее значение
2	bext-auto-time	Устанавливает время BEXT на текущее значение
3	bext-auto-date	Устанавливает дату BEXT на текущее значение
4	str-auto-date	Устанавливает дату строки на текущее значение

Таблица 7. Команда sndfile-play — список операционных систем, использующих соответствующие интерфейсы прикладного программирования.

Операционная система	API
MS Win32	waveOut[1]
Linux	ALSA or OSS[2]
MacOSX 10.6	CoreAudio[3]
MacOSX 10.7	AudioToolbox[4]
FreeBSD	/dev/dsp (OSS)[5]
OpenBSD	sndio[6]
Solaris	/dev/audio[7]

[1] Прим. автора. (Таран В.В.). Класс устройств wave/out состоит из аудиоустройств для низкоуровневого волнового аудиовыхода. Доступ к этим устройствам осуществляется с помощью волновых функций, описанных в Пакете разработки программного обеспечения платформы (SDK). Устройства этого класса связаны с линейными устройствами, которые поддерживают тип среды передачи LINEMEDIAMODE_AUTOMATEDVOICE, указанного в элементе dwMediaModes структуры LINEDEVCAPS для линейного устройства.

[2] Прим. автора. (Таран В.В.). Advanced Linux Sound Architecture (ALSA) — это программная платформа и часть ядра Linux, которая обеспечивает использование интерфейса прикладного программирования (API) для драйверов устройств звуковых карт. OSS-это альтернативная звуковая архитектура для Unix-подобных и POSIX-совместимых систем. Третья версия OSS была первоначальной звуковой системой для Linux, но затем в 2002 г. была заменена на Advanced Linux Sound Architecture (или ALSA), а 4 версия OSS стала проприетарным (закрытым) программным обеспечением. В 2007 г. когда 4Front Technologies выпустила свой исходный код и предоставила его под лицензией GPL, OSSv4 снова стал свободным программным обеспечением.

[3] Прим. автора. (Таран В.В.). Core Audio является низкоуровневым API для работы со звуком в операционных системах Apple macOS и iOS. Она включает в себя возможности кросс-платформенного OpenAL.

[4] Прим. автора. (Таран В.В.). Набор инструментов для качественных манипуляций со звуковыми данными. Позволяет записывать или воспроизводить аудио, конвертировать форматы, анализировать аудиопотоки и настраивать аудиосессию. Платформа AudioToolbox предоставляет интерфейсы для записи, воспроизведения и анализа потоков. В iOS платформа предоставляет дополнительные интерфейсы для управления аудиосессиями. API предназначен для использования традиционных Unix- подобных систем open(), read(), write () и ioctl () с помощью специальных устройств. Например, устройство по умолчанию для ввода и вывода звука — /dev/dsp.

[5] Прим. автора. (Таран В.В.). OSS-это альтернативная звуковая архитектура для Unix-подобных и POSIX-совместимых систем. см сноску 11. /dev/dsp — интерфейсы вывода звука в Unix-подобных операционных системах.

[6] Прим. автора. (Таран В.В.). sndio — это программный уровень операционной системы OpenBSD, который управляет звуковыми картами и MIDI-портами. Он предоставляет дополнительный звуковой сервер и документированный интерфейс прикладного программирования для единообразного доступа к серверу или аудио- и MIDI-оборудованию. sndio предназначен для работы с настольными приложениями, но уделяет особое внимание механизмам синхронизации и надежности, требуемым музыкальными приложениями. Основным компонентом sndio является sndiod audio и MIDI-сервер. Он блокирует разрыв между программными требованиями и оборудованием без программного обеспечения, представленным драйверами устройств операционной системы. Сервер позволяет выполнять множество полезных функций, среди которых:

- Смешивание и маршрутизация аудиосигнала нескольких программ — это позволяет нескольким программам одновременно использовать аудиоустройство.
- Разделение аудиоустройства на подустройства, например, разрешив одной программе использовать передние динамики, а другой — задние динамики, поскольку они являются независимыми простыми стереоустройствами.
- Маршрутизация аудио — и MIDI данных по сети которая позволяет программам, работающим на одном компьютере, использовать звуковую карту другого компьютера.

[7] Прим. автора. (Таран В.В.). /dev/audio (также как и /dev/dsp) — основной файл устройств для цифровых приложений. Любые данные, записанные в эти файлы, воспроизводятся на DAC/PCM/DSP устройстве звуковой карты. Чтение из этих файлов возвращает звуковые данные, записанные с текущего входного источника (по умолчанию таким источником служит микрофонный вход).

Рис. 3. Цикл установки библиотеки libsndfile в операционной системе MS Windows.

Таблица 8. Список команд, предназначенных для управления звуком в библиотеке libsndfile.

КОМАНДЫ БИБЛИОТЕКИ LIBSNDFILE		
№	**Команды**	**Расшифровка команды (кратко)**
	SFC_GET_LIB_VERSION	Команда даёт возможность получения версии библиотеки.
	SFC_GET_LOG_INFO	Команда даёт возможность извлечения внутреннего журнала операций для каждого файла.
	SFC_CALC_SIGNAL_MAX	Команда позволяет вычислить измеренное максимальное значение сигнала.
	SFC_CALC_NORM_SIGNAL_MAX	Команда позволяет произвести вычисление измеренного нормированного максимального значения сигнала.
	SFC_CALC_MAX_ALL_CHANNELS	Команда даёт возможность расчёта пикового значения для каждого канала.
	SFC_CALC_NORM_MAX_ALL_CHANNELS	Команда считает нормированное пиковое значение для каждого канала.
	SFC_GET_SIGNAL_MAX	Команда извлекает пиковое значение для файла (так как хранится в заголовке файла).
	SFC_GET_MAX_ALL_CHANNELS	Команда даёт возможность получения пикового значения для каждого канала (так как хранится в заголовке в заголовке файла).
	SFC_SET_NORM_FLOAT	Команда изменяет характеристики нормализации функций чтения и записи с плавающей запятой.
	SFC_SET_NORM_DOUBLE	Команда изменяет характеристики нормализации функций чтения и записи с плавающей запятой двойной точности.
	SFC_GET_NORM_FLOAT	Команда получает текущие характеристики нормализации функций чтения и записи с плавающей запятой.
	SFC_GET_NORM_DOUBLE	Команда осуществляет получение текущей характеристики нормализации функций чтения и записи с плавающей запятой двойной точности.
	SFC_SET_SCALE_FLOAT_INT_READ	Команда устанавливает / убирает масштабный коэффициент, когда целочисленные (короткие/объединённые) данные считываются из файла, содержащего данные с плавающей запятой.
	SFC_SET_SCALE_INT_FLOAT_WRITE	Команда устанавливает / убирает масштабный коэффициент, когда целочисленные (короткие/объединённые) данные записываются в файл, как данные с плавающей запятой.
	SFC_GET_SIMPLE_FORMAT_COUNT	Команда позволяет получить число *простых форматов*, поддерживаемых файлом «libsnd».
	SFC_GET_SIMPLE_FORMAT	Команда получает информацию о *простом формате*.
	SFC_GET_FORMAT_INFO	Команда получает информацию о *главном* или *подтиповом* форматах.
	SFC_GET_FORMAT_MAJOR_COUNT	Команда получает число *главных форматов*.
	SFC_GET_FORMAT_MAJOR	Команда получает информацию о *главном типе формата*.
	SFC_GET_FORMAT_SUBTYPE_COUNT	Команда получает число *субформатов*.
	SFC_GET_FORMAT_SUBTYPE	Команда получает информацию о *субформате*.
	SFC_SET_ADD_PEAK_CHUNK	Команда переключает программный код для добавления фрагмента «ПИК» в файлы WAVE (.wav) и AIFF.
	SFC_UPDATE_HEADER_NOW	Команда позволяет использовать канал, когда файл открыт для записи (эта команда также обновит заголовок файла, чтобы отразить данные, записанные до настоящего времени).
	SFC_SET_UPDATE_HEADER_AUTO	Команда позволяет использовать канал, когда файл открыт для записи (эта команда приведет к обновлению заголовка файла после каждой записи в файл).
	SFC_FILE_TRUNCATE	Команда обрезает файл, открытый для записи или чтения / записи.
	SFC_SET_RAW_START_OFFSET	Команда изменяет смещение начала поступления данных для файлов, открытых как SF_FORMAT_RAW.
	SFC_SET_CLIPPING	Команда включает и выключает автоматическое ограничение при выполнении преобразования с плавающей запятой в целое число.
	SFC_GET_CLIPPING	Команда получает установку ограничителя.

Таблица 8 (продолжение). Список команд, предназначенных для управления звуком в библиотеке libsndfile.

КОМАНДЫ БИБЛИОТЕКИ LIBSNDFILE		
№	Команды	Расшифровка команды (кратко)
	SFC_GET_EMBED_FILE_INFO	Команда получает информацию об аудиофайлах, встроенных в другие файлы.
	SFC_GET_AMBISONIC	Команда проверяет звуковой файл формата объёма.
	SFC_SET_AMBISONIC	Команда изменяет заголовок WAVEX для формата объёма.
	SFC_SET_VBR_ENCODING_QUALITY	Команда устанавливает переменную скорость передачи данных для обеспечения качества кодирования.
	SFC_SET_COMPRESSION_LEVEL	Команда устанавливает уровень сжатия.
	SFC_RAW_NEEDS_ENDSWAP	Команда определяет, требуется ли конечная замена необработанных данных.
	SFC_GET_BROADCAST_INFO	Команда получает информацию о широковещательном фрагменте.
	SFC_SET_BROADCAST_INFO	Команда устанавливает широковещательный фрагмент для получения информации.
	SFC_SET_CART_INFO	Команда устанавливает часть фрагмента для получения информации.
	SFC_GET_CART_INFO	Команда получает информацию из части фрагмента.
	SFC_GET_LOOP_INFO	Команда получает информацию о цикле.
	SFC_GET_INSTRUMENT	Команда получает информацию об инструменте.
	SFC_SET_INSTRUMENT	Команда устанавливает получение информации об инструменте.
	SFC_GET_CUE_COUNT	Команда получает счёт маркера разметки
	SFC_GET_CUE	Команда получает информацию маркера разметки.
	SFC_SET_CUE	Команда устанавливает получение информации маркера разметки.
	SFC_RF64_AUTO_DOWNGRADE	Команда позволяет обеспечить автозагрузку из RF64 в WAVE.

В Unix-подобных операционных системах могут присутствовать дополнительные команды sndfile-interleave[1], sndfile-deinterleave[2], sndfile-salvage[3], sndfile-concat[4]. Эти команды предназначены для расширения традиционной подсистемы кодирования и декодирования аудиоданных в зависимости от выбранного пользователем дистрибутива операционной системы (рис. 3).

Интересен также и прогрессивный подход авторов данной разработки к процессам внедрения новых функций в алгоритмы библиотеки. Libsndfile имеет широкий набор тестов, позволяющих эффективно выявлять ошибки, благодаря чему свежие релизы от них практически застрахованы. К примеру, если обнаруживаются ошибки, то к набору существующих тестов добавляются новые тесты, которые выполняют функции якорей, обеспечивающих страховку от того, что ошибки не вернутся обратно в программный код релиза. При интеграции новых функций тесты добавляются в набор тестов для того, чтобы убедиться, что оригинальные функции (уже включая новые функции) продолжают работать корректно, даже если они считаются старыми функциями. Чтобы сформировать представления о командах данной библиотеки с целью их дальнейшего систематического анализа, а впоследствии их использования в прикладных целях, упорядочим команды в табличном виде (таблица 8).

Итак, мы видим, что библиотека libsndfile обладает довольно обширным перечнем инженерных (про-

[1] Прим. автора. (Таран В.В.). создает многоканальный файл, берущий аудиоданные из двух или более монофонических файлов в качестве отдельных каналов. Формат вывода файла определяется суффиксом его имени. Звуковые параметры выходного файла будут сделаны таким образом, чтобы формат мог вместить каждый из моно входов; например, частота дискретизации будет максимальной частотой дискретизации, происходящей на входах. Выходной файл будет перезаписан, если он уже существует.

[2] Прим. автора. (Таран В.В.). sndfile-deinterleave создает два или более монофайла из многоканального аудиофайла, содержащего данные с отдельных каналов. Названия результирующих монофайлов обозначаются как `name_XY.suf", где имя и расширение suf являются базовым именем и суффиксом исходного файла.Если какой-либо файл под таким названием уже существует он будет перезаписан. Помимо числа каналов аудиоформат результирующих монофонических файлов будет таким же, как и в исходном файле.

[3] Прим. автора. (Таран В.В.). Аудиофайлы, использующие хранилище файлов WAV/WAVE, по своей сути ограничены полями размера данных 4G в заголовке WAV, хранящимися в виде 32-х битных целых чисел в абсолютном выражении. Многие приложения испытывают проблемы с файлами WAV/WAVE, размерами более 4G. sndfile-salvage — перезаписывает WAV-файл в файл W64 с тем же аудиоконтентом. Файл перезаписывается, если оригинальная его версия уже существует.

[4] Прим. автора. (Таран В.В.). sndfile-concat создает новый выходной файл путем объединения аудиоданных двух или более входных файлов. Кодировка выходного файла — это кодировка, используемая в infile1. Аудиоданные из последующих файлов преобразуются в эту кодировку. Единственное ограничение заключается в том, что файлы должны иметь одинаковое количество каналов. Выходной файл перезаписывается, если он уже существует.

цедурных) команд — всего их 45. Теперь попытаемся представить данные команды в более подробном виде с некоторыми примерами их использования.

Первая команда, представленная в таблице **SFC_GET_LIB_VERSION**. Команда обеспечивает извлечение версии библиотеки в виде строки. Её параметры таковы:

sndfile: Not used (не используемый)
cmd: SFC_GET_LIB_VERSION
data: A pointer to a char buffer
datasize: The size of the buffer

Пример потенциального использования:
char buffer [128];
sf_command (NULL, SFC_GET_LIB_VERSION, buffer, sizeof (buffer));

Возвращаемое значение: Это действие возвращает длину полученной строки версии[1].

Вторая команда **SFC_GET_LOG_INFO.** Извлекает буфер журнала, созданного при открытии файла в виде строки. Зачастую буфер журнала может являться веской причиной, по которой libsndfile не удаётся открыть конкретный файл. Имеет следующие параметры:

sndfile: A valid SNDFILE* pointer
cmd: SFC_GET_LOG_INFO
data: A pointer to a char buffer
datasize: The size of the buffer

Пример потенциального использования:
char buffer [2048];
sf_command (sndfile, SFC_GET_LOG_INFO, buffer, sizeof (buffer));

Возвращаемое значение: Это действие вернёт длину полученной строки версии[2].

Третья команда **SFC_CALC_SIGNAL_MAX.** Производит извлечение уже измеренного максимального значения сигнала. Она включает в себя чтение всего файла, которое может быть довольно медленным на файлах с большим весом.
sndfile: A valid SNDFILE* pointer
cmd: SFC_CALC_SIGNAL_MAX
data: A pointer to a double
datasize: sizeof (double)

Пример потенциального использования:
double max_val;
sf_command (sndfile, SFC_CALC_SIGNAL_MAX, &max_val, sizeof (max_val));

Возвращаемое значение: Ноль в случае успешного действия, в противном случае — ненулевое значение.

Четвёртая команда **SFC_CALC_NORM_SIGNAL_MAX.** Извлекает уже измеренное нормализованное максимальное значение сигнала. Читает весь файл, в зависимости от объёма файла скорость чтения может быть ограничена.

sndfile: A valid SNDFILE* pointer
cmd: SFC_CALC_NORM_SIGNAL_MAX
data: A pointer to a double
datasize: sizeof (double)

Пример потенциального использования:
double max_val;
sf_command (sndfile, SFC_CALC_NORM_SIGNAL_MAX, &max_val, sizeof (max_val));

Возвращаемое значение: Ноль в случае успешного действия, в противном случае — ненулевое значение.

Пятая команда **SFC_CALC_MAX_ALL_CHANNELS.** Вычисляет пиковое значение (то есть одно число) для каждого канала. Она включает в себя чтение всего файла, которое может быть медленным, если файл будет большого размера.

sndfile: A valid SNDFILE* pointer
cmd: SFC_CALC_MAX_ALL_CHANNELS
data: A pointer to a double
datasize: sizeof (double)* number_of_channels

Пример потенциального использования:
double peaks [number_of_channels];
sf_command (sndfile, SFC_CALC_MAX_ALL_CHANNELS, peaks, sizeof (peaks));

Возвращаемое значение: Ноль, если пики были вычислены успешно, и ненулевые значения в противном случае.

Шестая команда **SFC_CALC_NORM_MAX_ALL_CHANNELS.** Организует расчёт нормализованного пика для каждого канала. Функция читает весь файл, однако при его большом объёме процесс чтения может быть существенно увеличен.

sndfile: A valid SNDFILE* pointer
cmd: SFC_CALC_NORM_MAX_ALL_CHANNELS

[1] Строка, возвращаемая в буфер, переданный этой функции, не будет переполнять буфер и всегда будет иметь нулевое завершение.

[2] Строка, возвращаемая в буфер, переданный этой функции, не будет переполнять буфер и всегда будет иметь нулевое завершение.

```
size_t   sf_read_float    (SNDFILE *sndfile, float *ptr, size_t items) ;
size_t   sf_readf_float   (SNDFILE *sndfile, float *ptr, size_t frames) ;

size_t   sf_write_float   (SNDFILE *sndfile, float *ptr, size_t items) ;
size_t   sf_writef_float  (SNDFILE *sndfile, float *ptr, size_t frames) ;
```

Рис. 4

data: A pointer to a double
datasize: sizeof (double) * number_of_channels

Пример потенциального использования:
double peaks [number_of_channels];
sf_command (sndfile, SFC_CALC_NORM_MAX_ALL_CHANNELS, peaks, sizeof (peaks));

Возвращаемое значение: Ноль, если пики были вычислены успешно, и ненулевые значения в противном случае.

Седьмая команда **SFC_GET_SIGNAL_MAX.** Функция предназначена для извлечения пикового значения для аудиосигнала, сохранённого в заголовке файла.

sndfile: A valid SNDFILE* pointer
cmd: SFC_GET_SIGNAL_MAX
data: A pointer to a double
datasize: sizeof (double)

Пример потенциального использования:
double max_peak;
sf_command (sndfile, SFC_GET_SIGNAL_MAX, &max_peak, sizeof (max_peak));

Значение возврата: SF_TRUE, если заголовок файла содержит пиковое значение. SF_FALSE в противном случае.

Восьмая команда **SFC_GET_MAX_ALL_CHANNELS.** Извлекает пиковое значение для аудиосигнала, сохраненного в заголовке файла.

sndfile: A valid SNDFILE* pointer
cmd: SFC_GET_SIGNAL_MAX
data: A pointer to an array of doubles
datasize: sizeof (double) * number_of_channels

Пример потенциального использования:
double peaks [number_of_channels];
sf_command (sndfile, SFC_GET_MAX_ALL_CHANNELS, peaks, sizeof (peaks));

Значение возврата: SF_TRUE, если заголовок файла содержит пиковые файловые значения для каждого канала. SF_FALSE в противном случае.

Девятая команда **SFC_SET_NORM_FLOAT.** Эта команда влияет только на данные, считываемые или записываемые с помощью функций с плавающей запятой (рис. 4).

Параметры команды
sndfile: A valid SNDFILE* pointer
cmd: SFC_SET_NORM_FLOAT
data: NULL
datasize: SF_TRUE or SF_FALSE

1. Для операций чтения установка нормализации на SF_TRUE означает, что данные всех последующих операций чтения будут нормализованы до диапазона [–1.0, 1.0].
2. Для операций записи установка нормализации в SF_TRUE означает, что все данные, подаваемые в функции записи с плавающей запятой, должны находиться в диапазоне [–1.0, 1.0] и будут масштабироваться для формата файла по мере необходимости.
3. В обоих случаях установка нормализации на SF_FALSE означает, что масштабирование не будет происходить.

Пример потенциального использования:
sf_command (sndfile, SFC_SET_NORM_FLOAT, NULL, SF_TRUE);
sf_command (sndfile, SFC_SET_NORM_FLOAT, NULL, SF_FALSE);

Возвращаемое значение: Возвращает предыдущий режим нормализации с плавающей запятой.

Десятая команда **SFC_SET_NORM_DOUBLE.** Эта команда влияет только на данные, считываемые или записываемые с помощью функций с плавающей запятой двойной точности (рис. 5).

Параметры команды
sndfile: A valid SNDFILE* pointer
cmd: SFC_SET_NORM_DOUBLE
data: NULL
datasize: SF_TRUE or SF_FALSE

1. Для операций чтения установка нормализации на SF_TRUE означает, что данные всех последующих операций чтения будут нормализованы до диапазона [–1.0, 1.0].

```
size_t   sf_read_double    (SNDFILE *sndfile, double *ptr, size_t items) ;
size_t   sf_readf_double   (SNDFILE *sndfile, double *ptr, size_t frames) ;

size_t   sf_write_double   (SNDFILE *sndfile, double *ptr, size_t items) ;
size_t   sf_writef_double  (SNDFILE *sndfile, double *ptr, size_t frames) ;
```

Рис. 5

2. Для операций записи установка нормализации на SF_TRUE означает, что все данные, подаваемые в функции двойной записи, должны находиться в диапазоне [–1.0, 1.0] и будут масштабироваться по формату файла по мере необходимости.
3. В обоих случаях установка нормализации на SF_FALSE означает, что масштабирование не будет происходить.

Пример потенциального использования:

sf_command (sndfile, SFC_SET_NORM_DOUBLE, NULL, SF_TRUE);

sf_command (sndfile, SFC_SET_NORM_DOUBLE, NULL, SF_FALSE);

Возвращаемое значение: Возвращает предыдущий режим двойной нормализации.

Одиннадцатая команда **SFC_GET_NORM_FLOAT.** Подразумевает извлечение текущего режима нормализации с плавающей запятой.

sndfile: A valid SNDFILE* pointer
cmd: SFC_GET_NORM_FLOAT
data: NULL
datasize: anything

Пример потенциального использования:

normalisation = sf_command (sndfile, SFC_GET_NORM_FLOAT, NULL, 0);

Возвращаемое значение: Возвращает TRUE, если нормализация включена, и FALSE в противном случае.

Двенадцатая команда **SFC_GET_NORM_DOUBLE.** Извлекает текущий режим нормализации поплавка.

sndfile: A valid SNDFILE* pointer
cmd: SFC_GET_NORM_DOUBLE
data: NULL
datasize: anything

Пример потенциального использования:

normalisation = sf_command (sndfile, SFC_GET_NORM_DOUBLE, NULL, 0);

Возвращаемое значение: Возвращает TRUE, если нормализация включена, и FALSE в противном случае.

Тринадцатая команда **SFC_SET_SCALE_FLOAT_INT_READ.** Установка / удаление коэффициента масштабирования, когда целочисленные (короткие/целочисленные) данные считываются из файла, содержащего данные с плавающей запятой.

sndfile: A valid SNDFILE* pointer
cmd: SFC_SET_SCALE_FLOAT_INT_READ
data: NULL
datasize: TRUE or FALSE

Пример потенциального использования:

sf_command (sndfile, SFC_SET_SCALE_FLOAT_INT_READ, NULL, SF_TRUE);

Возвращаемое значение: Возвращает предыдущую настройку **SFC_SET_SCALE_FLOAT_INT_READ** для этого файла.

Четырнадцатая команда **SFC_SET_SCALE_INT_FLOAT_WRITE.** Установка / удаление коэффициента масштабирования, когда целочисленные (короткие/целочисленные) данные записываются в файл в виде данных с плавающей запятой.

sndfile: A valid SNDFILE* pointer
cmd: SFC_SET_SCALE_FLOAT_INT_READ
data: NULL
datasize: TRUE or FALSE

Пример потенциального использования:

sf_command (sndfile, SFC_SET_SCALE_INT_FLOAT_WRITE, NULL, SF_TRUE);

Возвращаемое значение: Возвращает предыдущую настройку **SFC_SET_SCALE_INT_FLOAT_WRITE** для этого файла.

Пятнадцатая команда **SFC_GET_SIMPLE_FORMAT_COUNT.** Извлечение имеющихся простых форматов, поддерживаемых libsndfile.

sndfile: Not used.
cmd: SFC_GET_SIMPLE_FORMAT_COUNT
data: a pointer to an int
datasize: sizeof (int)

```
SF_FORMAT_INFO  format_info ;
int                k, count ;
     sf_command (sndfile, SFC_GET_SIMPLE_FORMAT_COUNT, &count, sizeof (int)) ;

     for (k = 0 ; k < count ; k++)
     {   format_info.format = k ;
         sf_command (sndfile, SFC_GET_SIMPLE_FORMAT, &format_info, sizeof
         (format_info)) ;
         printf ("%08x  %s %s\n", format_info.format, format_info.name,
         format_info.extension) ;
         } ;
```

Рис. 6

Пример потенциального использования:
int count;
sf_command (sndfile, SFC_GET_SIMPLE_FORMAT_COUNT, &count, sizeof (int));

Возвращаемое значение: 0

Шестнадцатая команда **SFC_GET_SIMPLE_FORMAT.** Извлечение информации о простом формате.

sndfile: Not used.
cmd: SFC_GET_SIMPLE_FORMAT
data: a pointer to an SF_FORMAT_INFO struct
datasize: sizeof (SF_FORMAT_INFO)

Структура **SF_FORMAT_INFO** определена в <sndfile.h> как:

typedef struct
{int format;
const char *name;
const char *extension;
} SF_FORMAT_INFO;

Когда sf_command () вызывается с помощью **SF_GET_SIMPLE_FORMAT**, значением поля формата должно быть число формата (т.е. 0 <= format <= count value, полученное с помощью **SF_GET_SIMPLE_FORMAT_COUNT**).

Пример потенциального использования: рис. 6.

1. Возвращаемое значение: 0 в случае успешного действия и ненулевое значение в противном случае.
2. Значение поля формата структуры SF_FORMAT_INFO будет значением, которое может быть помещено в поле формата структуры SF_INFO при открытии файла для записи.
3. Поле названия будет содержать указатель литеры (признака)* на имя строки, например «WAVE (Microsoft 16 bit PCM)».
4. Поле расширения будет содержать наиболее часто используемое расширение файла для этого типа файла.

Семнадцатая команда **SFC_GET_FORMAT_INFO**. Получение информации о главном или подтиповом формате.

sndfile: Not used.
cmd: SFC_GET_FORMAT_INFO
data: a pointer to an SF_FORMAT_INFO struct
datasize: sizeof (SF_FORMAT_INFO)

Структура SF_FORMAT_INFO определена в <sndfile.h> как:

typedef struct
{int format;
const char *name;
const char *extension;
} SF_FORMAT_INFO;

При вызове sf_command () с помощью **SF_GET_FORMAT_INFO** проверяется поле формата, и если (format & **SF_FORMAT_TYPEMASK**) является допустимым форматом, то структура заполняется информацией о данном основном типе. Если (format & **SF_FORMAT_TYPE MASK**) имеет значение FALSE и (format & **SF_FORMAT_SUBMASK**) является допустимым форматом подтипа, то структура заполняется информацией о данном подтипе (рис. 7).

Возвращаемое значение: 0 в случае успешного действия и ненулевое значение в противном случае.

Восемнадцатая команда **SFC_GET_FORMAT_MAJOR_COUNT**. Извлечение имеющихся основных форматов.

sndfile: Not used.
cmd: SFC_GET_FORMAT_MAJOR_COUNT
data: a pointer to an int
datasize: sizeof (int)

Пример потенциального использования:
int count;

```
SF_FORMAT_INFO  format_info ;
    format_info.format = SF_FORMAT_WAV ;
    sf_command (sndfile, SFC_GET_FORMAT_INFO, &format_info, sizeof
    (format_info)) ;
    printf ("%08x  %s %s\n", format_info.format, format_info.name,
    format_info.extension) ;

    format_info.format = SF_FORMAT_ULAW ;
    sf_command (sndfile, SFC_GET_FORMAT_INFO, &format_info, sizeof
    (format_info)) ;
    printf ("%08x  %s\n", format_info.format, format_info.name) ;
```

Рис. 7

```
SF_FORMAT_INFO  format_info ;
    int            k, count ;
    sf_command (sndfile, SFC_GET_FORMAT_MAJOR_COUNT, &count, sizeof (int)) ;
    for (k = 0 ; k < count ; k++)
    {   format_info.format = k ;
        sf_command (sndfile, SFC_GET_FORMAT_MAJOR, &format_info, sizeof
        (format_info)) ;
        printf ("%08x  %s %s\n", format_info.format, format_info.name,
        format_info.extension) ;
        } ;
```

Рис. 8

sf_command (sndfile, SFC_GET_FORMAT_MAJOR_COUNT, &count, sizeof (int));

Возвращаемое значение: 0

Девятнадцатая команда **SFC_GET_FORMAT_MAJOR**. Извлечение информации об основном типе формата.

sndfile: Not used.
cmd: SFC_GET_FORMAT_MAJOR
data: a pointer to an SF_FORMAT_INFO struct
datasize: sizeof (SF_FORMAT_INFO)

Пример потенциального использования:рис. 8.

1. Более полный пример см. в программе list_formats.c в каталоге примеры/директория дистрибутива исходного кода libsndfile. Возвращаемое значение: 0 в случае успешного действия и ненулевое значение в противном случае.
2. Значение поля формата будет одним из основных идентификаторов формата, таких как **SF_FORMAT_WAV** или **SF_FORMAT_AIFF**.
3. Поле названия будет содержать указатель «литера» (признак)* на имя строки, например «WAV/WAVE (Microsoft)».
4. Поле расширения будет содержать наиболее часто используемое расширение файла для этого типа файла.

Двадцатая команда **SFC_GET_FORMAT_SUBTYPE_COUNT**. Извлечение имеющихся субформатов

sndfile: Not used.
cmd: SFC_GET_FORMAT_SUBTYPE_COUNT
data: a pointer to an int
datasize: sizeof (int)

Пример потенциального использования:

int count;
sf_command (sndfile, SFC_GET_FORMAT_SUBTYPE_COUNT, &count, sizeof (int));

Возвращаемое значение: 0

Двадцать первая команда **SFC_GET_FORMAT_SUBTYPE.** Перечислите подтипы (эта функция не переводит подтип в строку, описывающую этот подтип). Типичным вариантом использования может быть получение строкового описания всех подтипов, чтобы можно было заполнить диалоговое окно.

```
SF_FORMAT_INFO  format_info ;
        int         k, count ;
        sf_command (sndfile, SFC_GET_FORMAT_SUBTYPE_COUNT, &count, sizeof (int)) ;
        for (k = 0 ; k < count ; k++)
        {   format_info.format = k ;
            sf_command (sndfile, SFC_GET_FORMAT_SUBTYPE, &format_info, sizeof
            (format_info)) ;
            if (! sf_format_check (format_info.format | SF_FORMAT_WAV))
            continue ;
            printf ("%08x  %s\n", format_info.format, format_info.name) ;
            } ;
```

Рис. 9

```
SF_FORMAT_INFO  format_info ;
        int         k, count ;
        sf_command (sndfile, SFC_GET_FORMAT_SUBTYPE_COUNT, &count, sizeof (int)) ;
        for (k = 0 ; k < count ; k++)
        {   format_info.format = k ;
            sf_command (sndfile, SFC_GET_FORMAT_SUBTYPE, &format_info, sizeof
            (format_info)) ;
            if (format_info.format == SF_FORMAT_PCM_16)
            {   printf ("%08x  %s\n", format_info.format, format_info.name) ;
                break ;
                } ;
            } ;
```

Рис. 10

sndfile: Not used.
cmd: SFC_GET_FORMAT_SUBTYPE
data: a pointer to an SF_FORMAT_INFO struct
datasize: sizeof (SF_FORMAT_INFO)

Пример 1. Извлечение всех подтипов, поддерживаемых форматом WAV (рис. 9).

Пример 2. Печать строки, описывающей подтип SF_FORMAT_PCM_16 (рис. 10).

1. Более полный пример см. в программе list_formats.c в каталоге примеры/ директория дистрибутива исходного кода libsndfile. Возвращаемое значение:
2. 0 в случае успешного действия и ненулевое значение в противном случае.
3. Значение поля формата будет одним из основных идентификаторов формата, таких как SF_FORMAT_WAV или SF_FORMAT_AIFF.
4. Поле названия будет содержать указатель литера (признак)* на имя строки, например «WAV (Microsoft) «или» AIFF (Apple/SGI)».
5. Поле расширения будет указателем NULL.

Двадцать вторая команда **SFC_SET_ADD_PEAK_CHUNK**. По умолчанию файлы WAV и AIFF, содержащие данные с плавающей запятой (подтип **SF_FORMAT_FLOAT** или **SF_FORMAT_DOUBLE**), имеют ПИКОВЫЙ фрагмент. С помощью этой команды[1] добавление ПИКОВОГО фрагмента может быть включено или выключено.

sndfile: A valid SNDFILE* pointer
cmd: SFC_SET_ADD_PEAK_CHUNK
data: Not used (should be NULL)
datasize: TRUE or FALSE.

Пример потенциального использования:рис. 11.

Возвращаемое значение:

1. Возвращает SF_TRUE, если пиковый фрагмент будет записан после этого вызова.
2. Возвращает SF_FALSE, если пиковый фрагмент не будет записан после этого вызова.

[1] Прим. автора. (Таран В.В.). Указанная команда должна быть исполнена до того, как какие-либо данные будут записаны в файл.

```
/* Turn on the PEAK chunk. */
        sf_command (sndfile, SFC_SET_ADD_PEAK_CHUNK, NULL, SF_TRUE) ;

        /* Turn off the PEAK chunk. */
        sf_command (sndfile, SFC_SET_ADD_PEAK_CHUNK, NULL, SF_FALSE) ;
```

Рис. 11

```
/* Turn on auto header update. */
        sf_command (sndfile, SFC_SET_UPDATE_HEADER_AUTO, NULL, SF_TRUE) ;

        /* Turn off auto header update. */
        sf_command (sndfile, SFC_SET_UPDATE_HEADER_AUTO, NULL, SF_FALSE) ;
```

Рис. 12

Двадцать третья команда **SFC_UPDATE_HEADER_NOW**. Заголовок аудиофайла обычно записывается с помощью libsndfile, когда файл закрывается с помощью sf_close (). Однако, существуют ситуации, когда генерируются большие файлы, и было бы неплохо иметь допустимые (достоверные) данные в заголовке до того, как файл будет завершен. Использование этой команды обновит заголовок файла, чтобы отразить объем данных, записанных в файл до сих пор. Другие программы, открывающие файл для чтения (до того, как будут записаны дополнительные данные), затем считывают допустимый заголовок звукового файла.

sndfile: A valid SNDFILE* pointer
cmd: SFC_UPDATE_HEADER_NOW
data: Not used (should be NULL)
datasize: Not used.

Пример потенциального использования:

/* Update the header now. */
sf_command (sndfile, SFC_UPDATE_HEADER_NOW, NULL, 0);

Возвращаемое значение: 0.

Двадцать четвёртая команда **SFC_SET_UPDATE_HEADER_AUTO**. Аналогичен SFC_UPDATE_HEADER_NOW, но обновляет заголовок в конце каждого вызова функций sf_write*.

sndfile: A valid SNDFILE* pointer
cmd: SFC_UPDATE_HEADER_NOW
data: Not used (should be NULL)
datasize: SF_TRUE or SF_FALSE

Пример потенциального использования: рис. 12.

Возвращаемое значение — TRUE, если заголовок автоматического обновления включен; FALSE в противном случае.

Двадцать пятая команда **SFC_FILE_TRUNCATE**. Усечение файла, открытого для записи или чтения/записи.

sndfile: A valid SNDFILE* pointer
cmd: SFC_FILE_TRUNCATE
data: A pointer to an sf_count_t.
datasize: sizeof (sf_count_t)

Обрежьте файл до числа кадров, определенного параметром sf_count_t, на который указывают данные. После этой команды указатель чтения и записи будет находиться в новом конце файла. Эта команда завершится неудачей (возвращается ненулевое значение), если запрошенная позиция усечения находится за пределами конца файла.

Пример потенциального использования: рис. 13.

Возвращаемое значение: Ноль в случае успешного действия, ненулевое значение в противном случае.

Двадцать шестая команда **SFC_SET_RAW_START_OFFSET**. Изменение начального смещения данных для файлов, открытых как SF_FORMAT_RAW.

sndfile: A valid SNDFILE* pointer
cmd: SFC_SET_RAW_START_OFFSET
data: A pointer to an sf_count_t.

```
/* Truncate the file to a length of 20 frames. */
        sf_count_t  frames = 20 ;
        sf_command (sndfile, SFC_FILE_TRUNCATE, &frames, sizeof (frames)) ;
```

Рис. 13

```
/  Reset the data offset to 5 bytes from the start of the file. */
sf_count_t  offset = 5 ;
sf_command (sndfile, SFC_SET_RAW_START_OFFSET, &offset, sizeof (offset)) ;
```

Рис. 14

datasize: sizeof (sf_count_t)

Для файла, открытого в формате F_FORMAT_RAW, установите смещение данных в соответствии с заданным значением.

Пример потенциального использования: рис. 14.

Возвращаемое значение: Ноль при успешном действии, ненулевое значение в противном случае.

Двадцать седьмая команда **SFC_SET_CLIPPING**. Включение / выключение автоматического отсечения при преобразовании числа с плавающей запятой в целое число.

sndfile: A valid SNDFILE* pointer
cmd: SFC_SET_CLIPPING
data: NULL
datasize: SF_TRUE or SF_FALSE.

Включение (data size == SF_TRUE) или выключение (datasize == SF_FALSE) отсечения.

Пример потенциального использования:
sf_command(sndfile, SFC_SET_CLIPPING, NULL, SF_TRUE);

Возвращаемое значение: Режим отсечения (SF_TRUE или SF_FALSE).

Двадцать восьмая команда **SFC_GET_CLIPPING**. Включение / выключение автоматического отсечения при преобразовании числа с плавающей запятой в целое число.

sndfile: A valid SNDFILE* pointer
cmd: SFC_GET_CLIPPING
data: NULL
datasize: 0

Извлечение текущей настройки отсечения

ПРИМЕР ПОТЕНЦИАЛЬНОГО ИСПОЛЬЗОВАНИЯ:

sf_command (sndfile, SFC_GET_CLIPPING, NULL, 0);

Возвращаемое значение: В режиме отсечения (SF_TRUE или SF_FALSE).

Двадцать девятая команда **SFC_GET_EMBED_FILE_INFO.** Определить смещение файла и длину файла, встроенного в другой более крупный файл.

sndfile: A valid SNDFILE* pointer
cmd: SFC_GET_CLIPPING
data: a pointer to an SF_EMBED_FILE_INFO struct
datasize: sizeof (SF_EMBED_FILE_INFO)

Структура SF_EMBED_FILE_INFO определена в <sndfile.h> как:

typedef struct
{sf_count_t offset;
sf_count_t length;
} SF_EMBED_FILE_INFO;

1. Возвращаемое значение: 0 в случае успешного действия и ненулевое значение в противном случае.
2. Значением поля смещения структуры **SF_EMBED_FILE_INF**O будут смещения в байтах от начала внешнего файла до начала аудиофайла.
3. Значением поля смещения структуры **SF_EMBED_FILE_INFO** будет длина встроенного файла в байтах.

Тридцатая команда **SFC_GET_AMBISONIC**. Проверка, в случае если текущий файл имеет GUID файла WAVEX[1] для любого из форматов, Объёмного звука.

sndfile: A valid SNDFILE* pointer

[1] Форматы Ambisonic WAVEX доступны по адресу: http://dream.cs.bath.ac.uk/researchdev/wave-ex/bformat.html.

cmd: SFC_WAVEX_GET_AMBISONIC
data: NULL
datasize: 0

Возвращаемое значение: **SF_AMBISONIC_NONE** или SF_AMBISONIC_B_FORMAT или ноль, если формат файла не поддерживает форматы объемного звука.

Тридцать первая команда **SFC_SET_AMBISONIC**. Установка GUID из нового файла WAVEX, чтобы указать формат Объемного звука.

sndfile: A valid SNDFILE* pointer
cmd: SFC_WAVEX_SET_AMBISONIC
data: NULL
datasize: SF_AMBISONIC_NONE or SF_AMBISONIC_B_FORMAT

1. Включить (SF_AMBISONIC_B_FORMAT) или выключить (SF_AMBISONIC_NONE) кодировку. В настоящее время эта команда поддерживается только для файлов с форматом SF_FORMAT_WAVE[1].
2. Возвращаемое значение: Возвращает только что установленное значение объемного звука или нулевое значение, если формат файла не поддерживает кодировку объемного звука.

Тридцать вторая команда **SFC_SET_VBR_ENCODING_QUALITY.** Установите качество кодирования Переменной Скорости Передачи Битов. Значение качества кодирования должно быть между 0.0 (самое низкое качество) и 1.0 (самое высокое качество). В настоящее время эта команда реализована только для файлов FLAC и Ogg/Vorbis. Это никак не влияет на несжатые форматы файлов[2].

sndfile: A valid SNDFILE* pointer
cmd: SFC_SET_VBR_ENCODING_QUALITY
data: A pointer to a double value
datasize: sizeof (double)

Возвращаемое значение: SF_TRUE, если было задано качество кодирования VBR. SF_FALSE в противном случае.

Тридцать третья команда **SFC_SET_COMPRESSION_LEVEL**. Установка уровня сжатия. Уровень сжатия должен быть между 0.0 (минимальный уровень сжатия) и 1.0 (самый высокий уровень сжатия). В настоящее время эта команда реализована только для файлов FLAC и Ogg/Vorbis. Это никак не влияет на несжатые форматы файлов[3].

sndfile: A valid SNDFILE* pointer
cmd: SFC_SET_COMPRESSION_LEVEL
data: A pointer to a double value
datasize: sizeof (double)

Возвращаемое значение: SF_TRUE, если был установлен уровень сжатия. SF_FALSE в противном случае.

Тридцать четвёртая команда **SFC_RAW_NEEDS_ENDSWAP.** Определить, должны ли необработанные данные, считанные с помощью sf_read_raw, быть полностью заменены на главном процессоре.

Например, вернется ли SF_TRUE при чтении WAV, содержащего сведения о SF_FORMAT_PCM_16 на большой платформе с определенным порядком следования байтов и SF_FALSE на небольшой платформе с определенным порядком следования байтов.

sndfile: A valid SNDFILE* pointer
cmd: SFC_RAW_NEEDS_ENDSWAP
data: NULL
datasize: 0

Возвращаемое значение: SF_TRUE или SF_FALSE.

Тридцать пятая команда **SFC_GET_BROADCAST_INFO**. Извлечение ФРАГМЕНТА ШИРОКОВЕЩАТЕЛЬНОГО РАСШИРЕНИЯ из WAV — файлов и связанных с ними файлов.

sndfile: A valid SNDFILE* pointer
cmd: SFC_GET_BROADCAST_INFO
data: a pointer to an SF_BROADCAST_INFO struct
datasize: sizeof (SF_BROADCAST_INFO)

Структура SF_BROADCAST_INFO определена в <sndfile.h> как рис. 14.

Возвращаемое значение: SF_TRUE, если файл содержит фрагмент Широковещательного Расширения, или SF_FALSE в противном случае.

Тридцать шестая команда **SFC_SET_BROADCAST_INFO.** Установка ФРАГМЕНТА ШИРОКОВЕЩАТЕЛНОГО РАСШИРЕНИЯ для WAV-файлов и связанных с ними файлов.

sndfile: A valid SNDFILE* pointer

[1] Прим. автора. (Таран В.В.). Форматы Ambisonic WAVEX определены здесь: http://dream.cs.bath.ac.uk/researchdev/wave-ex/bformat.html.

[2] Прим. автора. (Таран В.В.). Команда должна быть отправлена до того, как какие-либо аудиоданные будут записаны в файл.

[3] Прим. автора. (Таран В.В.). Команда должна быть отправлена до того, как какие-либо аудиоданные будут записаны в файл.

```
typedef struct
    {   char            description [256] ;
        char            originator [32] ;
        char            originator_reference [32] ;
        char            origination_date [10] ;
        char            origination_time [8] ;
        unsigned int    time_reference_low ;
        unsigned int    time_reference_high ;
        short           version ;
        char            umid [64] ;
        char            reserved [190] ;
        unsigned int    coding_history_size ;
        char            coding_history [256] ;
    } SF_BROADCAST_INFO ;
```

Рис. 14

```
#define SF_CART_INFO_VAR(p_tag_text_size) \
    struct
            {  char            version [4] ;
               char            title [64] ;
               char            artist [64] ;
               char            cut_id [64] ;
               char            client_id [64] ;
               char            category [64] ;
               char            classification [64] ;
               char            out_cue [64] ;
               char            start_date [10] ;
               char            start_time [8] ;
               char            end_date [10] ;
               char            end_time [8] ;
               char            producer_app_id [64] ;
               char            producer_app_version [64] ;
               char            user_def [64] ;
               long    level_reference ;
                       SF_CART_TIMER   post_timers [8] ;
               char            reserved [276] ;
               char            url [1024] ;
                       unsigned int        tag_text_size ;
               char            tag_text[p_tag_text_size] ;
            }
```

Рис. 15

cmd: SFC_SET_BROADCAST_INFO
data: a pointer to an SF_BROADCAST_INFO struct
datasize: sizeof (SF_BROADCAST_INFO)

Возвращаемое значение: SF_TRUE, если установка фрагмента Широковещательного Расширения была успешной, и SF_FALSE в противном случае.

Тридцать седьмая команда SFC_SET_CART_INFO. Извлечение ФРАГМЕНТА КОРЗИНЫ из WAV-файлов (и связанных с ними файлов). На основе стандарта AES46 для CartChunk (см. CartChunk.org для получения дополнительной информации).

sndfile: A valid SNDFILE* pointer

```
typedef struct
    {   short    time_sig_num ;   /* any positive integer    > 0  */
        short    time_sig_den ;   /* any positive power of 2 > 0  */
        int         loop_mode ;   /* see SF_LOOP enum             */

        int         num_beats ;   /* this is NOT the amount of quarter notes !!!*/
                                  /* a full bar of 4/4 is 4 beats */
                                  /* a full bar of 7/8 is 7 beats */

        float             bpm ;   /* suggestion, as it can be calculated using
        other fields:*/
                                  /* file's length, file's sampleRate and our
                                  time_sig_den*/
                                  /* -> bpms are always the amount of _quarter
                                  notes_ per minute */

        int    root_key ;         /* MIDI note, or -1 for None */
        int future [6] ;
    } SF_LOOP_INFO ;
```

Рис. 16

cmd: SFC_GET_CART_INFO
data: a pointer to an SF_CART_INFO struct
datasize: sizeof (SF_CART_INFO)

Структура SF_CART_INFO определена в <sndfile.h> как рис. 15.

Возвращаемое значение: SF_TRUE, если файл содержал фрагмент КОРЗИНЫ или SF_FALSE в противном случае.

Тридцать восьмая команда **SFC_GET_CART_INFO.** Установите ФРАГМЕНТ КОРЗИНЫ для WAVE-файлов (и связанных с ними файлов).

sndfile: A valid SNDFILE* pointer
cmd: SFC_SET_CART_INFO
data: a pointer to an SF_CART_INFO struct
datasize: sizeof (SF_CART_INFO)

Возвращаемое значение: SF_TRUE, если установка фрагмента КОРЗИНЫ была успешной, и SF_FALSE в противном случае.

Тридцать девятая команда **SFC_GET_LOOP_INFO.** Извлечение информации о цикле для файла, включая время подписи, длину биений и исходную базовую ноту MIDI.

sndfile: A valid SNDFILE* pointer
cmd: SFC_GET_LOOP_INFO
data: a pointer to an SF_LOOP_INFO struct
datasize: sizeof (SF_LOOP_INFO)

Структура **SF_BROADCAST_INFO** определена в <sndfile.h> как рис. 16.

Пример потенциального использования:
SF_LOOP_INFO loop;
sf_command (sndfile, SFC_GET_LOOP_INFO, &loop, sizeof (loop));

Возвращаемое значение:
1. SF_TRUE — если заголовок файла содержит информацию о цикле для файла.
2. SF_FALSE — в противном случае.

Сороковая команда **SFC_GET_INSTRUMENT**. Извлечение инструментальной информации из файла, включая базовую ноту MIDI, отображение клавиатуры и циклическую информацию (начало/стоп и режим).

sndfile: A valid SNDFILE* pointer
cmd: SFC_GET_INSTRUMENT
data: a pointer to an SF_INSTRUMENT struct

datasize: sizeof (SF_INSTRUMENT)

Структура **SF_INSTRUMENT** определена в <sndfile.h> как рис. 17.

Пример потенциального использования:
SF_INSTRUMENT inst;

```
enum
 {    /*
      ** The loop mode field in SF_INSTRUMENT will be one of the following.
      */
      SF_LOOP_NONE = 800,
      SF_LOOP_FORWARD,
      SF_LOOP_BACKWARD,
      SF_LOOP_ALTERNATING
 } ;

 typedef struct
 {   int gain ;
      char basenote, detune ;
      char velocity_lo, velocity_hi ;
      char key_lo, key_hi ;
      int loop_count ;

      struct
      {   int mode ;
          unsigned int start ;
          unsigned int end ;
          unsigned int count ;
      } loops [16] ; /* make variable in a sensible way */
 } SF_INSTRUMENT ;
```

Рис. 17

sf_command (sndfile, SFC_GET_INSTRUMENT, &inst, sizeof (inst));

Возвращаемое значение: SF_TRUE, если заголовок файла содержит инструментальную информацию для файла. SF_FALSE в противном случае.

Сорок первая команда **SFC_SET_INSTRUMENT**. Установка инструментальной информации для файла.

sndfile: A valid SNDFILE* pointer
cmd: SFC_SET_INSTRUMENT
data: a pointer to an SF_INSTRUMENT struct
datasize: sizeof (SF_INSTRUMENT)

Пример потенциального использования:
SF_INSTRUMENT inst;
sf_command (sndfile, SFC_SET_INSTRUMENT, &inst, sizeof (inst));

Возвращаемое значение: SF_TRUE, если заголовок файла содержит информацию об инструменте для файла. SF_FALSE в противном случае.

Сорок вторая команда **SFC_GET_CUE_COUNT**. Извлечение имеющихся монтажных маркеров доступных для получения посредством команды **SFC_GET_CUE**.

sndfile: A valid SNDFILE* pointer
cmd: SFC_GET_CUE
data: a pointer to a uint32_t
datasize: sizeof (uint32_t)

Пример потенциального использования:
uint32_t cue_count;
sf_command (sndfile, SFC_GET_CUE_COUNT, &cue_count, sizeof (cue_count));

Возвращаемое значение: SF_TRUE, если заголовок файла содержит информацию о монтажном маркере для файла. SF_FALSE в противном случае.

Сорок третья команда **SFC_GET_CUE**. Извлечение информации о монтажном маркере из файла.

sndfile: A valid SNDFILE* pointer
cmd: SFC_GET_CUE
data: a pointer to an SF_CUES struct
datasize: sizeof (SF_CUES)

Структура SF_CUES определена в <sndfile.h> как рис. 18.

Существует также SF_CUES_VAR #define, который позволяет читать/писать более 100 монтажных маркеров.

Пример потенциального использования:
SF_CUES cues;

```
typedef struct
    {   int cue_count ;

        struct
        {   int32_t   indx ;
            uint32_t  position ;
            int32_t   fcc_chunk ;
            int32_t   chunk_start ;
            int32_t   block_start ;
            uint32_t  sample_offset ;
            char name [256] ;
        } cue_points [100] ;
    } SF_CUES ;
```

Рис. 18

sf_command (sndfile, SFC_GET_CUE, &cues, sizeof (cues));

Возвращаемое значение: SF_TRUE, если заголовок файла содержит информацию о монтажном маркере для файла. SF_FALSE в противном случае.

Сорок четвёртая команда SFC_SET_CUE. Установите информацию о монтажном маркере для файла.

sndfile: A valid SNDFILE* pointer
cmd: SFC_SET_CUE
data: a pointer to an SF_CUES struct
datasize: sizeof (SF_CUES)

Пример потенциального использования:
SF_CUES cues;
sf_command (sndfile, SFC_SET_CUE, &cues, sizeof (cues));

Возвращаемое значение: SF_TRUE, если заголовок файла содержит информацию о монтажном маркере для файла. SF_FALSE в противном случае.

Сорок пятая команда **SFC_RF64_AUTO_DOWNGRADE.** Обеспечивает автоматическое понижение с RF64 до WAV. Рекомендация Европейского вещательного союза (**E**uropean **B**roadcasting **U**nion, EBU) заключается в том, что при записи файлов RF64 и результирующем файле размером менее 4Gig, его следует понизить до WAVE-файла (WAVE-файлы имеют максимальный размер 4Gig). Libsndfile не совсем точно следует рекомендациям EBU, главным образом потому, что тестовый набор должен иметь возможность проверки чтение/запись файлов RF64 без необходимости генерирования файлов размером более 4 гигабайт[1].

sndfile: A valid SNDFILE* pointer
cmd: SFC_RF64_AUTO_DOWNGRADE
data: NULL
datasize: SF_TRUE or SF_FALSE

Пример потенциального использования:
/* Enable auto downgrade on file close. */
sf_command (sndfile, SFC_RF64_AUTO_DOWNGRADE, NULL, SF_TRUE);

Возвращаемое значение:
1. **SF_TRUE** — если установлен **SFC_RF64_AUTO_DOWNGRADE**.
2. **SF_FALSE** — в противном случае.

Таким образом, мы рассмотрели наиболее важные команды библиотеки libsndfile. Хочется подчеркнуть, что библиотека libsndfile имеет лояльную и понятную схему лицензирования, что существенно упрощает разработчикам её наладку и подключение к собственным программным продуктам. Библиотека в виде посредника уже хорошо зарекомендовала себя при кодировании и декодировании аудиоформатов в таких программных гигантах как Audacity® и Adobe Audition, что характеризует её как надёжное вспомогательное звено в архитектуре любого аудиоредактора.

[1] Прим. автора. (Таран В.В.). Указанная команда должна быть отдана до того, как первый бит аудиоданных будет записан в файл. Вызов этой команды после записи аудиоданных вернёт текущее значение этого параметра, но не позволит его изменить.

Основной целью нашего мини-исследования являлся анализ базового функционала библиотеки и распространение знания по её использованию.

Ввиду недостаточности научных источников на русском языке по данной библиотеке, авторы надеются, что анализируемый ими материал откроет дальнейшие перспективы для научных исследований в данном направлении, а прикладные свойства, описанного в статье феномена, смогут послужить хорошим арсеналом средств при проектировании автономных программных приложений, нацеленных на качественную обработку звука.

ЛИТЕРАТУРА

1. Официальный сайт библиотеки libsndfile — http://www.mega-nerd.com/libsndfile/#SeeAlso [дата обращения к источнику: 23.04.2021].
2. Сайт на хостинге программных проектов — https://github.com/libsndfile/libsndfile [дата обращения к источнику: 23.04.2021].
3. Справочная информация — https://formulae.brew.sh/formula/libsndfile [дата обращения к источнику: 23.04.2021].
4. Справочная информация — https://stackoverflow.com/questions/tagged/libsndfile [дата обращения к источнику: 23.04.2021].
5. Справочная информация — https://www.freshports.org/audio/libsndfile [дата обращения к источнику: 23.04.2021].
6. Справочная информация —www.coderoad.ru/list/?page=1&sort=view&tag=libsndfile [дата обращения к источнику: 23.04.2021].
7. Чтение wav-файла с использованием libsndfile в C — https://progi.pro/chteniewav-fayla-s-ispolzovaniem-libsndfile-v-c-11930716 [дата обращения к источнику: 23.04.2021].

Журнал «Современная наука: актуальные проблемы теории и практики»

Всероссийский институт научной и технической информации РАН