Оглавление

[Описание задания 2](#_Toc15229967)

[1. Запись сообщения в файл 3](#_Toc15229968)

[Структура WAV-файла 3](#_Toc15229969)

[Структура сообщения 4](#_Toc15229970)

[Демонстрация работы 5](#_Toc15229971)

[2. Расшифровка 9](#_Toc15229972)

[Описание 9](#_Toc15229973)

[2.1 Вариант плагина без выбора расположения результата дешифрации 10](#_Toc15229974)

[2.2 Вариант плагина с выбором расположения 11](#_Toc15229975)

[Для работы плагина необходимо 14](#_Toc15229976)

[Ссылки на использованные ресурсы 15](#_Toc15229977)

# Описание задания

1. Разработать ПО для командной строки, осуществляющее стеганографическое встраивание в музыкальный файл.

* Язык разработки – любой
* Тип музыкального файла - любой
* Алгоритм встраивания – LSB
* Плотность встраивания (объем встаиваемых бит на байт) – задается параметром
* Путь к стегоконтейнеру – задается параметром
* Путь к файлу – задается параметром
* Расширение и размер встроенного файла должны храниться в стегоконтейнере произвольным образом
* В случае, если при заданных параметрах файл не может быть встроен должна выдаться ошибка
* Предусмотреть возможность тестирования стегоконтейнера. Т.е. на вход подается музыкальный файл, на выходе – возможные размеры файла при разной плотности встраивания

2. Разработать/Модифицировать кодек/модуль/плагин для популярного плеера (не VLC), осуществляющий извлечение данных из стегоконтейнера и воспроизведение музыкального файла

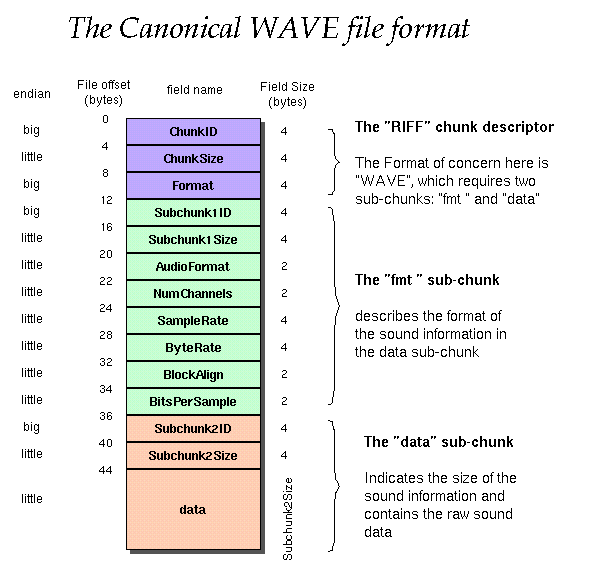
* Путь к папке, в которую будут извлекаться файлы из стеганоконтейнера указывается на этапе сборки/компиляции

# Запись сообщения в файл

Язык разработки: С++

Тип музыкального файла: WAV

## Структура WAV-файла



От 0 до 43 байта: служебная информация, которая не подлежит изменению.

От 44 байта до конца файла: аудиоданные, которые будут изменены.

Во-первых, нам потребуется значение поля «Subchunk2Size», равное количеству байт занятому под блок «data». Сам блок «data» состоит из сэмплов.

Во-вторых, для нахождения младших разрядов нам также понадобится значение поля «BitsPerSample», равное количеству бит на один сэмпл.

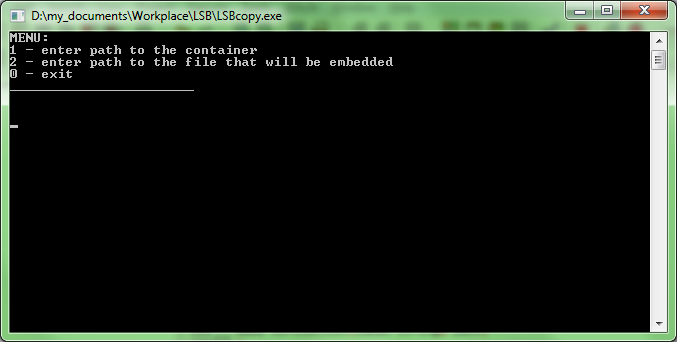
Согласно формулировке задания, пользователь может выбирать плотность встраивания. Выбор будет ограничен диапазоном значений: степень двойки, от 1 до «BitsPerSample».

## Структура сообщения

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| density (плотность встраивания) | mark (метка) | dataSize (размер сообщения) | fileName (имя файла и его расширение) | data (содержание сообщения) |
| - Количетсво бит в сэмпле, используемых для записи сообщения  - Значение задается 5 битами. Записывается изначально уменьшенным на единицу, то есть, к примеру, при значении 5, реальное значение плотности будет 6  - Запись производится в младший разряд первых пяти сэмплов. К примеру, сэмпл: хххх хххх станет хххх ххх1  - Извлекается в первую очередь | - Некоторое число, используемое для проверки возможности декодирования сообщения.  Если извлеченные с полученной плотностью 4 байта будут содержать число “111111”, то программа продолжит декодирование сообщения, в противном случае – остановка  - Занимает 4 байта | - Число, равное размеру сообщения в байтах  - Занимает 4 байта | - Строка, содержащая, имя файла и разрешение  - Занимает 20 байт | - Некоторые данные  - Занимает dataSize – 4 – 4 – 20 байт |

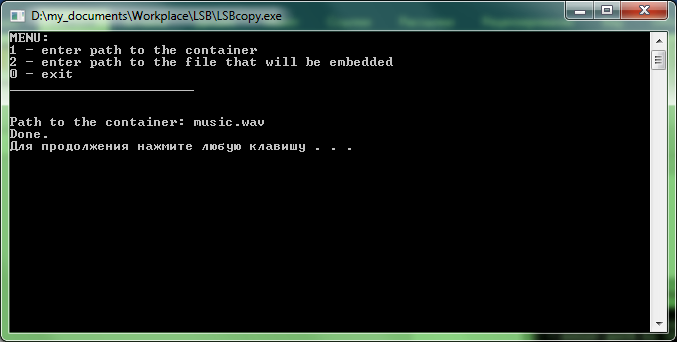
Все данные, кроме значения density, извлекаются из заданного количества младших бит сэмплов.

## Демонстрация работы

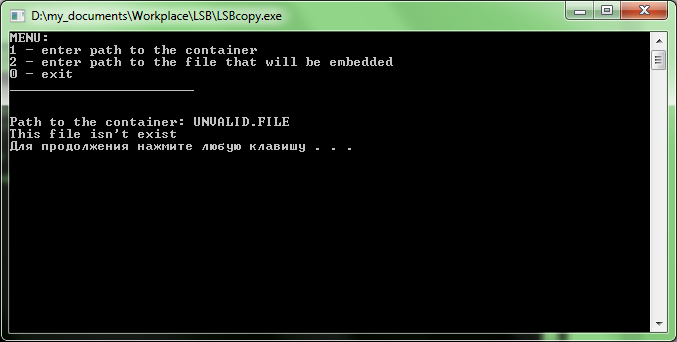


Программа требует от пользователя ввести расположение контейнера и расположение файла, который будет встраиваться. Пока программа не убедится в том, что данные введены и файлы существуют, процесс стеганографии будет заблокирован.

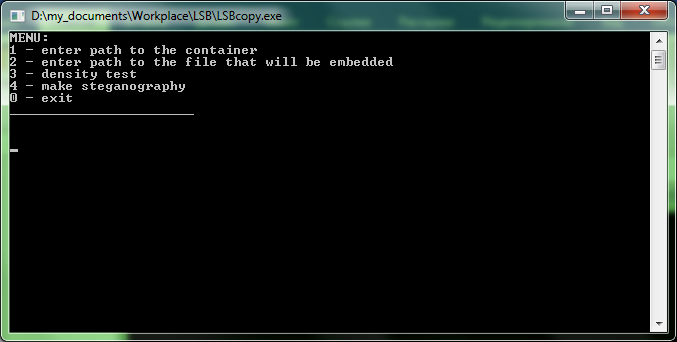
Для ввода данных необходимо нажать клавишу, соответствующую цифре напротив пункта меню. К примеру, нажатие «1» запустит процесс ввода адреса расположения контейнера и чтения данных из него.

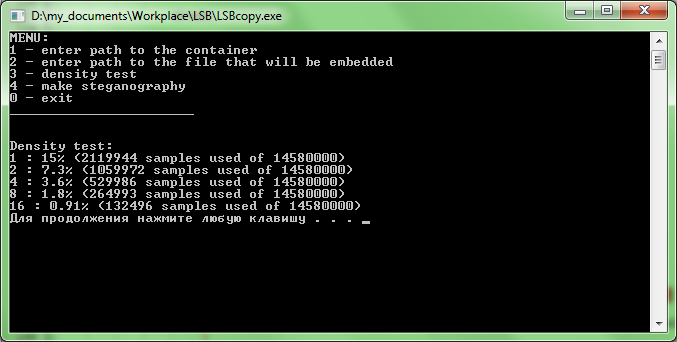


Если файл существует, программа выведет в консоль сообщение «Done». В противном случае, «This file isn’t exist».



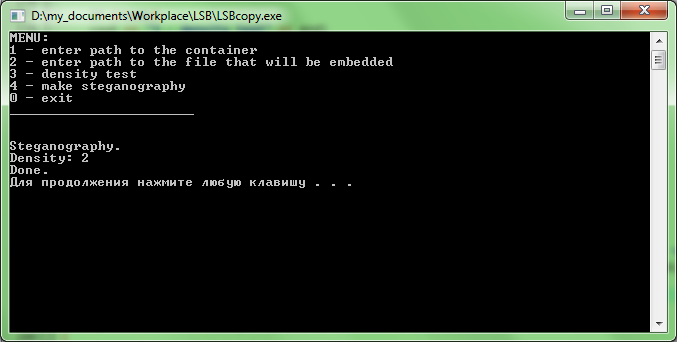
Далее, по завершению ввода пути к контейнеру и к файлу для записи, в меню открывается еще два пункта: тест глубины и непосредственно сам процесс стеганографии.

Записывать на трек будем картинку размером 88 269 байт.



Касательно теста глубины. К примеру, запись 1 бита файла на каждый сэмпл трека приведет к тому, что искажены будут примерно 2 миллиона сэмплов, что составляет 15% от всего трека.

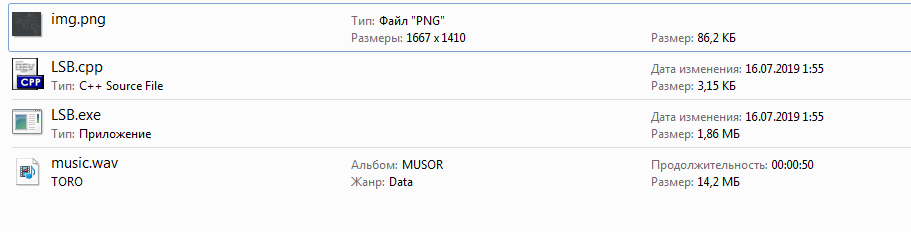
Тестирование необходимо проводить перед запуском стеганографического процесса, чтобы выбрать значение глубины. Чем меньше глубина, тем меньше различим накладываемый шум. Оптимально выбирать самое малое значение глубины, при этом такое, чтобы количество искаженных сэмплов не превышало их общего числа.



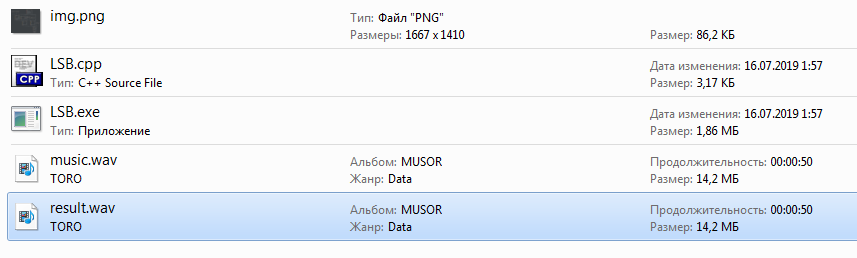
При запуске процесса стеганографии пользователь выбирает глубину.

По завершению процесса, будет выведено сообщение «Done».

Папка с программой до запуска программы



Папка с программой после окончания работы программы (результирующий файл появляется в папке с программой)

 Можно убедиться, что размер файла, а также длительность песни не увеличились.

# Расшифровка

Плеер: foobar2000

Разрабатываем: плагин

## Описание

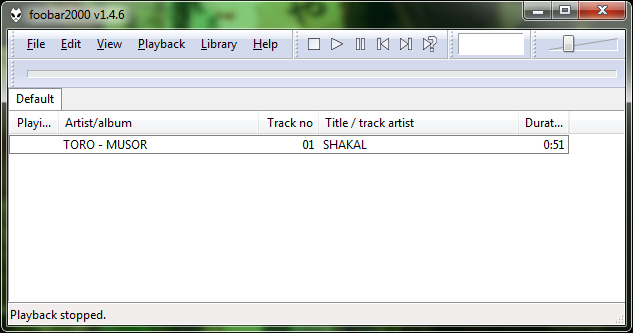
Плагины для плеера foobar2000 представляют собой dll-файлы, написанные с использованием SDK. Они хранятся в специальной папке components и подключаются при запуске программы.

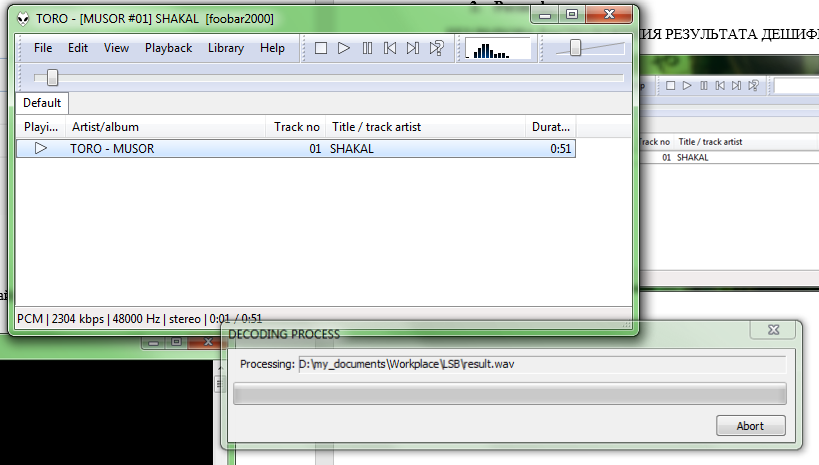
В нашем случае, необходимо обеспечить отслеживание события «открытие нового трека», чтобы запускать на фоне его декодирование. Оно осуществляется при помощи запуска Lua-скрипта, поэтому плагин, по большей степени, будет осуществлять лишь роль посредника, передающего в нужный момент информацию о треке. Для защиты от прерывания работы плеера используется отдельный поток, а также механизм мьютексов для обеспечения потока монопольным владением файлом.

Плагин передает расположение файла Lua-скрипту, который, в свою очередь, считывает побайтово файл, извлекает плотность, с ее помощью извлекает метку возможности декодирования, сверяет полученную метку с заданной внутри и, в случае совпадения, продолжает последовательно извлекать размер файла, название и расширение файла, а после и само его содержимое.

## 2.1 Вариант плагина без выбора расположения результата дешифрации

(путь задается в Lua-файле, по умолчанию: “D:\\new”)

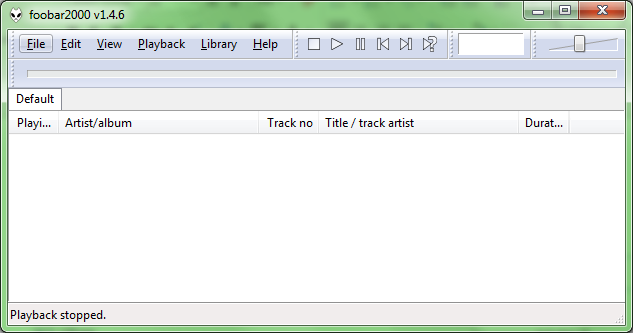
  
С началом проигрывания трека (мы отлавливаем событие «открытие нового трека») запускается отдельный поток, расшифровывающий содержимое стегоконтейнера и захватывающий мьютекс на время работы с файлом. Для защиты от декодирования файлов других форматов, перед началом производится проверка расширения файла.



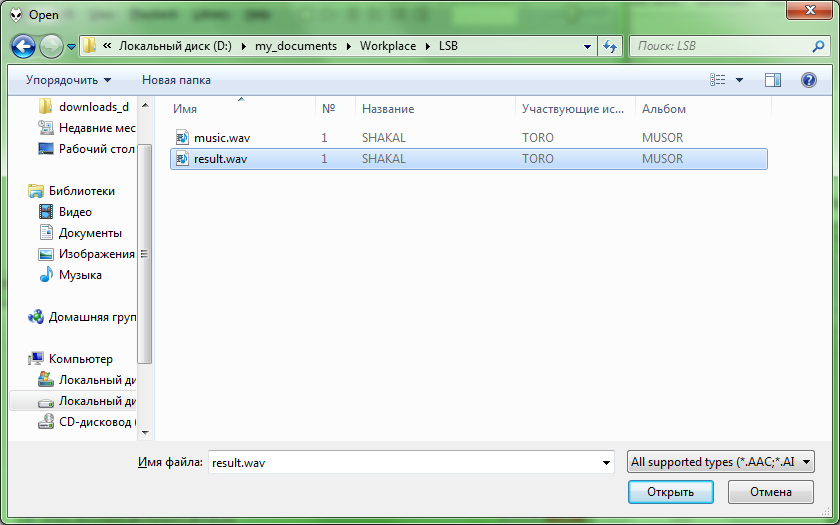
О том, что ведется работа над файлом оповещает всплывающее окно, которое пропадет по завершению работы.

## 2.2 Вариант плагина с выбором расположения

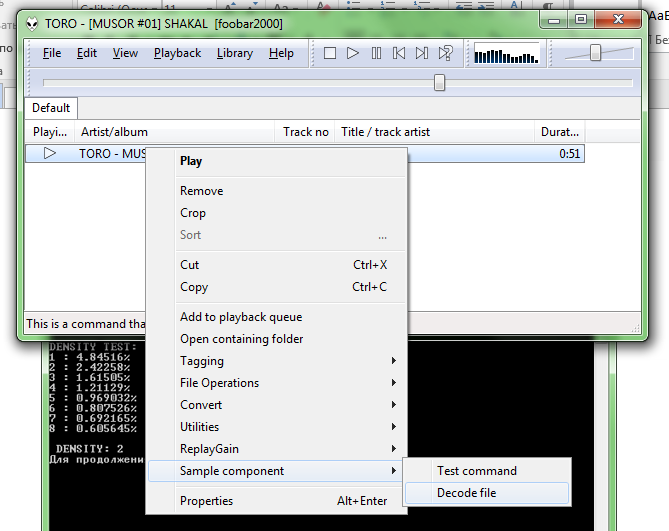
Графический интерфейс выбранного для модификации плеера.



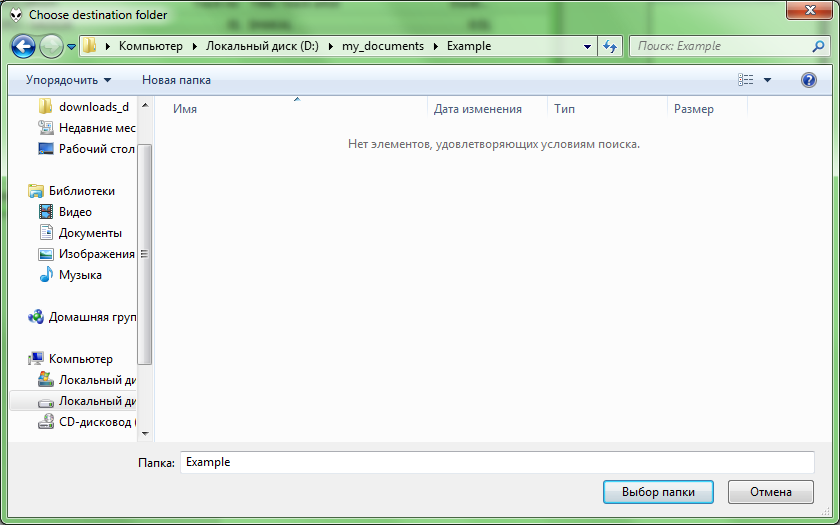
Выбран модифицированный трек.



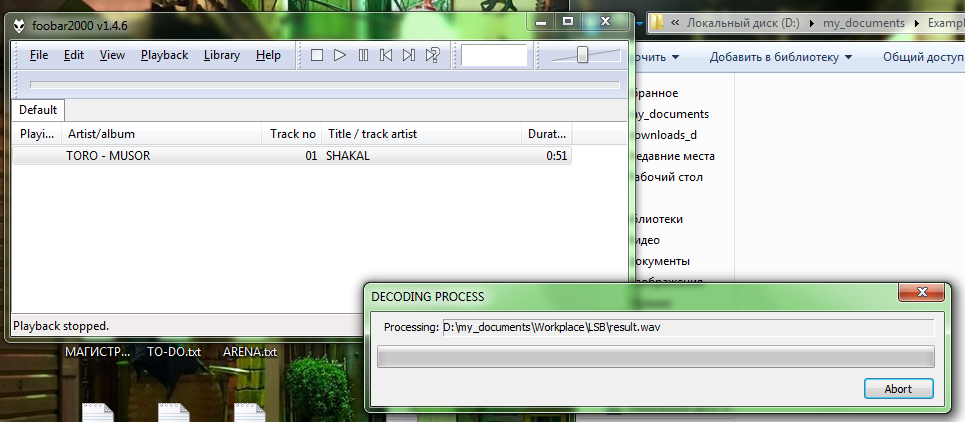
Щелчком правой кнопки мыши по треку выводится контекстное меню. В нем выбираем наш плагин «Sample Component» и команду «Decode file».



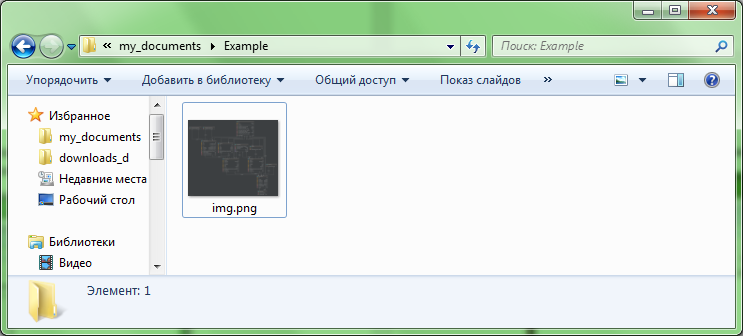
Далее появляется окно выбора расположения декодированного файла



После выбора расположения начинается процесс декодирования

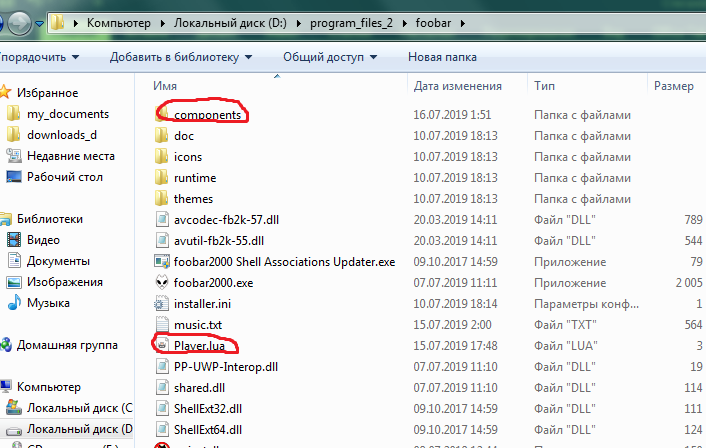


Результат работы программы



# Для работы плагина необходимо

1. Скачать плеер foobar2000
2. Копировать foo\_sample.dll в папку components к другим плагинам
3. Копировать файл Player.lua в папку с программой



# Ссылки на использованные ресурсы

1. Статья «**Структура WAV файла»**

URL: <https://audiocoding.ru/articles/2008-05-22-wav-file-structure/>

2. Статья «**Learn Lua in 15 Minutes»**

URL: <http://tylerneylon.com/a/learn-lua/>

3. Раздел форума, посвященный разработке плагинов для плеера foobar2000

URL: http://foobar2000.ru/forum/viewforum.php?f=2