Лабораторная работа №2

Использование библиотек

Инструментарий и требования к работе

Работа выполняется на C (C11 и новее). На сервере сборка под C17.

Задание

Необходимо написать программу, которая позволит найти временное смещение одного аудио-файла относительно другого методом кросс-корреляции (cross-correlation).

Аргументы программе передаются через командную строку в одном из следующих форматов:

#	Формат	Пояснение
1	`<file>`	Один двухканальный файл. Если аудио-каналов не 2, то необходимо сообщить о неверных данных и завершиться с ненулевым кодом.
2	`<file1> <file2>`	Два файла. Для обработки берётся первый канал из каждого файла.

Для работы с данными (например, чтения аудио-файлов) необходимо использовать библиотеку ffmpeg. Аудио-файлы могут быть следующих форматов:

1. FLAC (Free Lossless Audio Codec)
2. MP2 (MPEG audio layer 2)
3. MP3 (MPEG audio layer 3)
4. Opus
5. AAC (Advanced Audio Coding)

Разница выводится в миллисекундах относительно первого файла/канала. Если второй файл/канал позже первого, то значение положительно, иначе – отрицательно.

Если аудиопотоков в файле несколько, то берётся первый найденный.

Помимо найденной разницы необходимо вывести sample rate. Если у входных файлов sample rate разный, то:

- либо необходимо завершиться с сообщением о том, что это не поддерживается (и ненулевым кодом возврата),
- либо выполнить передискретизацию в больший sample rate.

Формат вывода: `"delta: %i samples\nsample rate: %i Hz\ndelta time: %i ms\n"`

Пример:

```
delta: 5644800 samples

sample rate: 44100 Hz

delta time: 128000 ms
```

Кроме ffmpeg разрешено также использовать библиотеку fftw. Другие сторонние библиотеки использовать запрещено (стандартную библиотеку C использовать можно и нужно).

Библиотека	**Версия**	**Подключение (пример)**
ffmpeg	6.1	`#include <libavformat/avformat.h>`
fftw	3.3	`#include <fftw3.h>`

Для вашего репозитория (на github) исходный файл, содержащий функцию main, должен лежать в корне репозитория.

Внимание! В репозитории не должно быть любых файлов ffmpeg и fftw. Вы можете их хранить рядом локально, но в этом случае они должны быть занесены `.gitignore`, а лучше в `.git/info/exclude`.

Пример:

```
/<github_repo_name>
├── main.c /* your src file */
├── my_src.c /* [optional] your src file */
├── src /* [optional] source dir */
    ├── my_src_file1.c /* [optional] your src file */
    ├── my_src_file2.c /* [optional] your src file */
├── my_header.h /* [optional] your header file */
├── include /* [optional] header dir */
    ├── my_header_file1.h /* [optional] your hdr file */
    ├── my_header_file2.h /* [optional] your hdr file */
├── .gitignore
├── .clang-format
├── ...
```

Должно присутствовать разумное разделение на файлы и правильно сформированные заголовочные файлы с минимальными зависимостями.

Если на вход программе поданы некорректные данные, то необходимо завершаться с человекочитаемым сообщением о возникшей ошибке (по-английски) и правильным кодом из `return_codes.h`.

Ссылки/материалы

- ☑ Download FFmpeg – ссылка на скачивание, нужны shared исходники.
- ☑ FFTW Download Page – ссылка на скачивание.
- ☑ https://skkv-itmo.gitbook.io/ct-c-cpp-course/build/building-program
- ☑ https://skkv-itmo.gitbook.io/ct-c-cpp-course/build/build-systems
- ☑ Компиляция из командной строки: https://www.youtube.com/watch?v=t2iHVBZdhdA

- ☑ Подключение внешних библиотек из командной строки: https://www.youtube.com/watch?v=5w3CZvsanKE
- ☑ Системы сборки и подключение внешних библиотек: https://skkv-itmo.gitbook.io/ct-c-cpp-course/build/libs
- ☑ Поиск утечек памяти: https://skkv-itmo.gitbook.io/ct-c-cpp-course/code-analysis/memory-leaks
- ☑ Санитайзеры: https://skkv-itmo.gitbook.io/ct-c-cpp-course/code-analysis/sanitizer
- ☑ Антипаттерны кода (рекомендуется посмотреть 3, 6, 7, 8, 11): https://skkv-itmo.gitbook.io/ct-c-cpp-course/best-practices/best-practices

Защита

На защите помимо объяснения своего кода и как он у вас собирается будет необходимо продемонстрировать навыки сборки вашего кода на стороннем компьютере.

Также принимающий может дать задание на защите, заключающееся в модификации кода.

Репозиторий

Если что-то не работает в репозитории и вы не понимаете почему – пишем Виктории или вашему проверяющему (если есть).

Известные проблемы:

1. Если после первого запуска `BuildTest` вы видите сообщение `"Init repo failed"`, то пишите Виктории с указанием ошибки и ссылкой на репозиторий. Эта ситуация возникает в случае, если по (пока неведанной нам) причине автоматически не запускается `Init` workflow при создании репозитория (ветки `main`). Если вы сразу видите, что у вас после взятия репозитория не было ни одного успешного запуска `Init`

(есть только один неуспешный), то действовать на опережение и сразу Виктории об этой проблеме.

2. В репозиториях, созданных до 04.04 устаревший **git-подмодуль suite**, использующийся для запуска локальной тестовой системы, с критическим багом — неправильное число аргументов в конструкторе объекта. Поэтому при запуске в результате теста может быть выведено следующее:

```
===> FAILED
====> ERROR: unknown.
[stderr]: TestResult.__init__() missing 2 required positional arguments: 'categories' and 'timer'
```

Если такая ошибка произошла, то это значит, что ваша программа на данном входе вернула ненулевое значение (SUCCESS). Для того, чтобы применить патч к тестовой системе необходимо:

   a. склонировать репозиторий рекурсивно: `git clone --recursive <url>`
   b. зайти в директорию `<repo>/suite/`
   c. обновить подмодуль suite: `git pull origin main`
   d. обновить подмодуль на GH – для этого создайте коммит: `git add suite/ && git commit -m "Updated submodule." && git push`