FFMPEG в приложениях .NET

для работы с медиафайлами

Андриевский Леонид
Ростелеком
Telegram:@slenik

FFMPEG и где он применяется

Это набор библиотек с открытым исходным кодом, которые позволяют записывать, конвертировать и транслировать (стримить) цифровые аудио- и видеоматериалы в различных форматах.

Точно используется	На моём ПК есть библиотеки ffmpeg в составе	
VLC media player	Edge	Skype
OBS-Studio	Slack	Mattermost
Blender	Postman	Microsoft Teams
HandBrake	Steam (Windows клиент)	Draw.io
VirtualDub2	Discord	WhatsApp

Каков план действий?

- Изучить терминологию.
- Понять, где взять библиотеки ffmpeg с нужными компонентами.
- Разобраться в том, как вызывать методы из библиотек ffmpeg.
- Написать кроссплатформенную утилиту с использованием ffmpeg, делающую кое-что полезное.

Терминология: контейнер

- **Контейнер** – формат файла или формат передачи данных, чьи спецификации определяют только способ представления данных в пределах одного файла.
- **Медиапоток** (или просто **поток**) – временная последовательность данных, закодированная в битовый поток.
- **Мультиплексирование** (muxing) – объединение нескольких медиапотоков в один контейнер. **Демультиплексирование** (demuxing) – разделение контейнера на несколько медиапотоков.

Терминология: кодек

- **Кодирование/декодирование** - процесс преобразования данных из одного формата в другой с некоторой целью.
  В рамках текущего доклада упоминаются методы кодирования и декодирования, цель которых – максимально уменьшить размер закодированных данных.
- **Кодек** (КОдировщик-ДЕКодировщик) - программа, способная выполнять кодирование и декодирование.

Контейнер (формат)	Тип	Поддерживает в том числе алгоритмы кодирования
Zip	Общий	LZW, Deflate, LZMA, PPMd
Bmp	Изображение	RGB, RLE, PNG
Png	Изображение	PNG, PNG+Deflate
mkv	Видео	Полный список доступен в wiki, не влезет в эту таблицу

Способы работы с FFMPEG

Через параметры командной строки

- в .NET реализуется очень просто – Process.Start,
- при некорректном использовании – падение только запущенного процесса, но не нашего ПО.
- Прогресс выполнения операции доступен через анализ вывода утилиты
- Основная часть кода = вызов ffmpeg с нужными параметрами

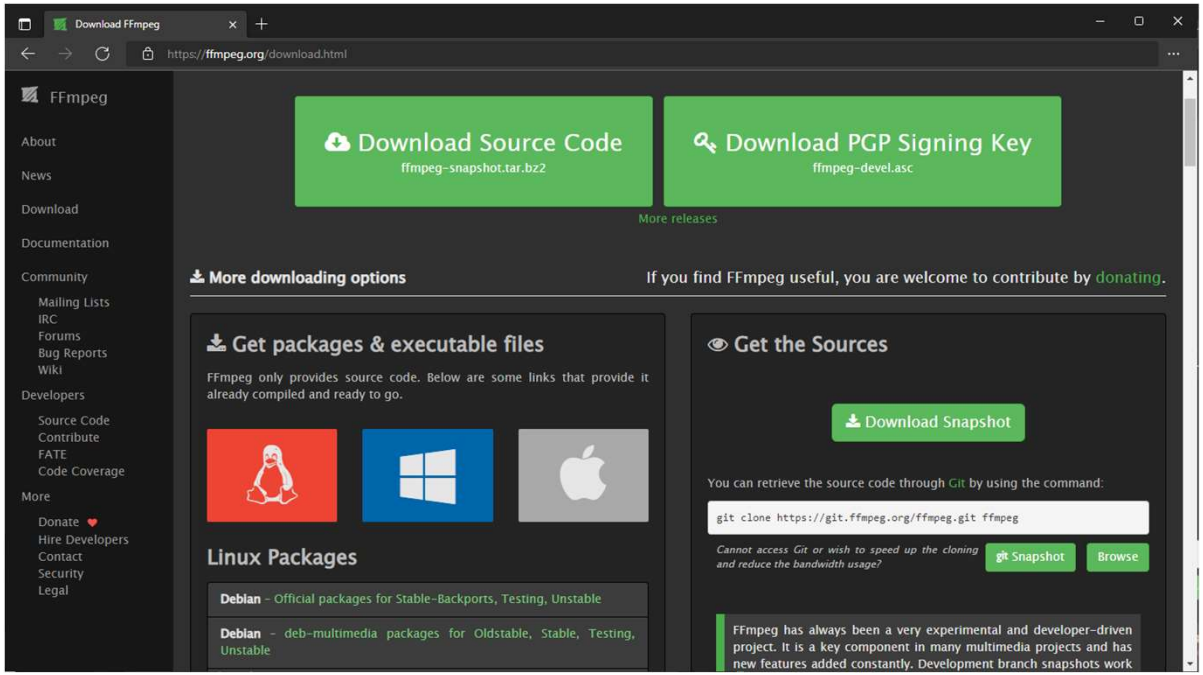
Через вызовы API библиотек

- в .NET реализуется через p/invoke, а также через сторонние пакеты (например, FFmpeg.AutoGen),
- при некорректном использовании – падение всего нашего приложения (или потребуется выносить части приложения в отдельные процессы).
- Прогресс выполнения операции зачастую легко отследить.
- Может потребоваться написать больше кода, чем при работе через командную строку

Сборка FFMPEG: берём готовую

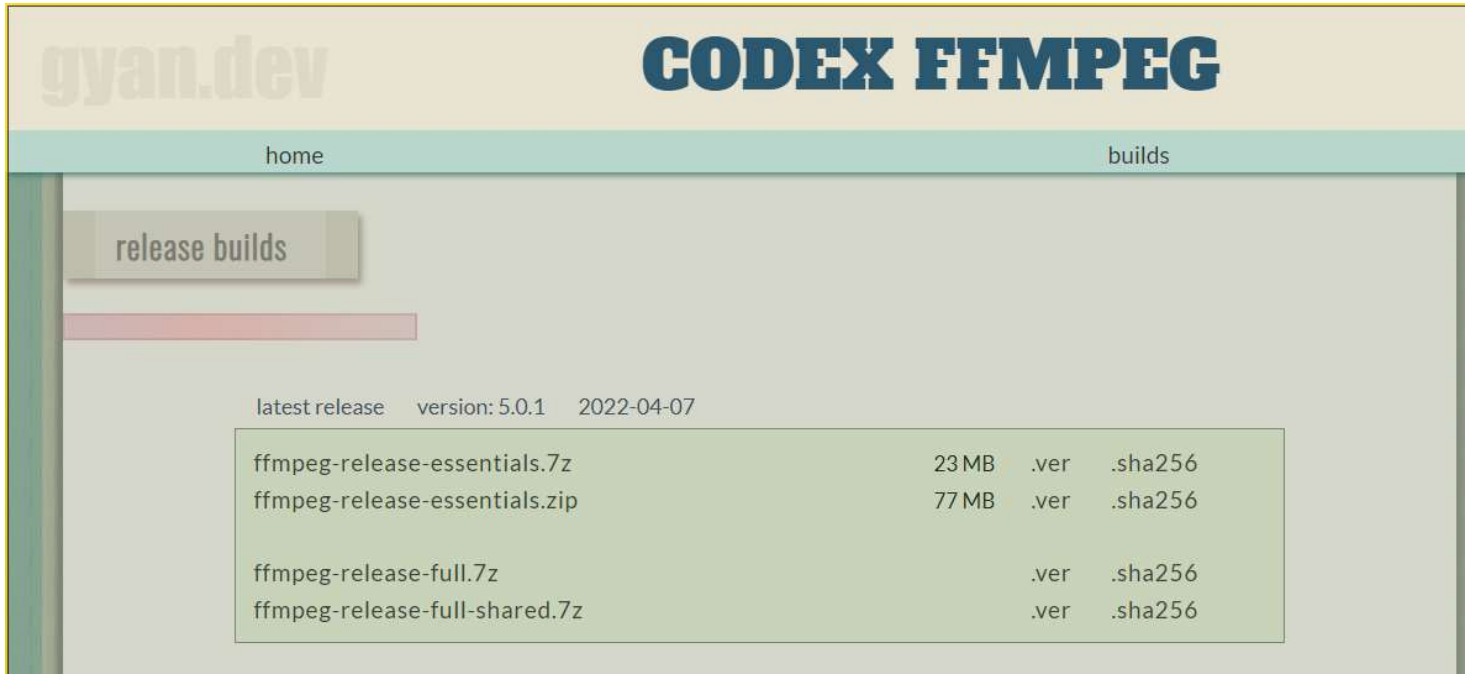
Для себя или для разработки – можно скачать уже скомпилированную

- Официальная страница ffmpeg
- ffmpeg-builds на github
- Сборки ffmpeg от Zeranoe (остались доступны лишь через web.archive.org)

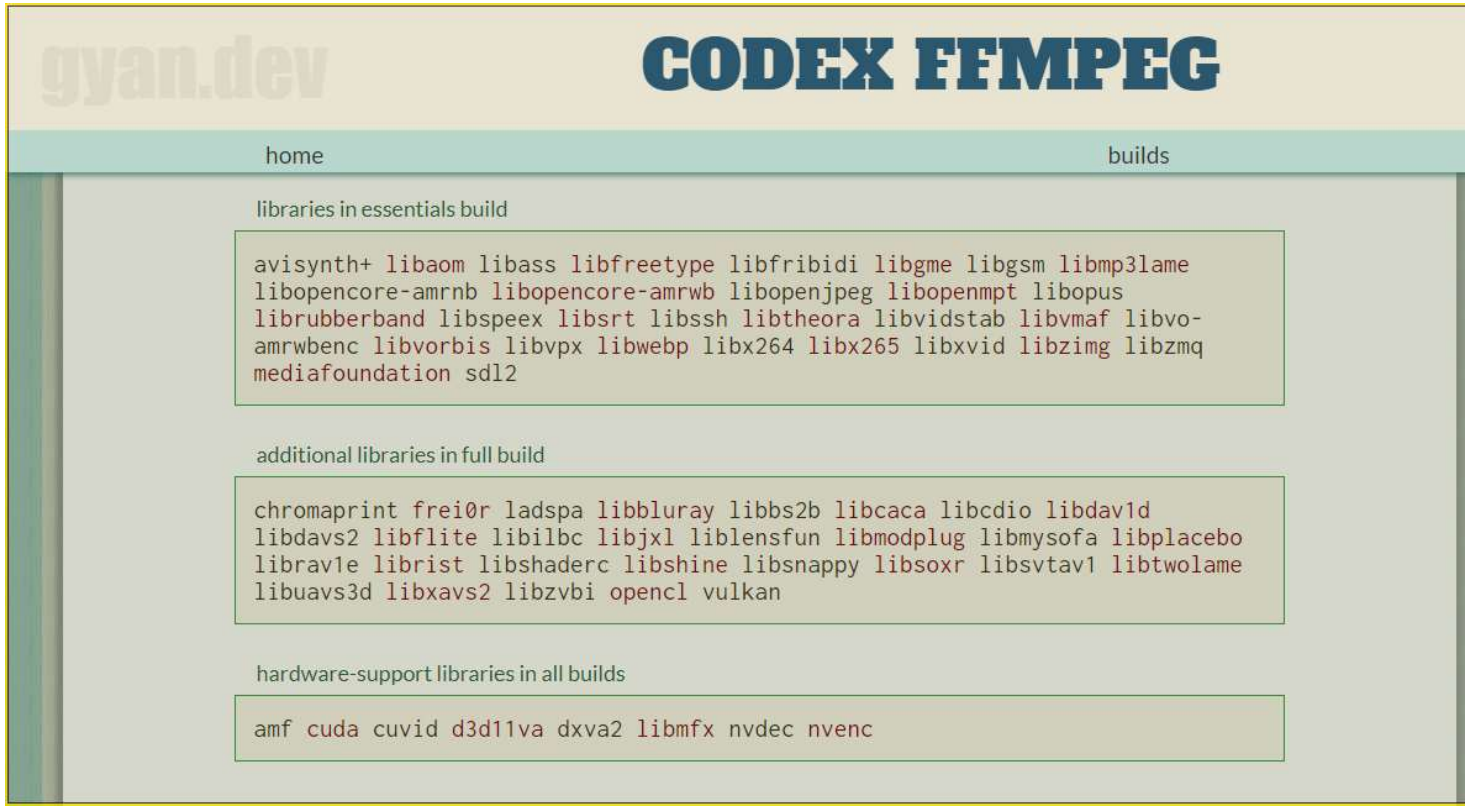


Обязательно проверяйте список добавленных в сборку библиотек!

Сборка FFMPEG: берём готовую



Сборка FFMPEG: берём готовую



Сборка FFMPEG: компилируем сами

Если FFMPEG требуются для распространения в составе ПО, то лучше скомпилировать свою сборку

– более 100 параметров,
– множество доступных зависимостей,
– требуется учитывать лицензии зависимых библиотек,
– могут быть использованы разные компиляторы (gcc, msvc),
+ полученная сборка будет максимально подходить по возможностям и кодекам.

- FFmpeg-Builds на github
- Media-autobuild_suite на github
- vcpkg от Microsoft

Промежуточные итоги

На текущий момент мы:

- Ознакомились с терминологией (контейнер, медиапоток, кодек).
- Узнали об основных возможностях фреймворка ffmpeg.
- Разобрались с вариантами получения сборки ffmpeg с нужными нам функциями.

Пример: собираем ffmpeg для утилиты работы с медиапотоками

Параметры для компиляции с указанием лицензий

- https://www.ffmpeg.org/general.html
- https://www.ffmpeg.org/legal.html

Выбираем:

- **Тип сборки**: **static** / shared
- **Работать со сборкой будем**: **через вызовы API** / через командную строку

- **Лицензия**: nonfree, GPLv3, **LGPLv3**, GPLv2.1, LGPLv2.1.
- **Кодеки**: отключаем все
- **Компоненты мультиплексирования**: включаем все, кроме платных

Пример: собираем ffmpeg для утилиты работы с медиапотоками

В результате получаем команду конфигурации (для сборки под Windows, скажем, компилятором msvc):

./configure --prefix=/c/ffmpeg --toolchain=msvc \

--enable-asm --enable-yasm --arch=i386 \

--disable-swscale --disable-doc --disable-ffplay --disable-ffprobe --enable-ffmpeg --enable-shared --disable-static \

--enable-libxml2 --enable-chromaprint --enable-avisynth --enable-libgme --enable-libmodplug --enable-vapoursynth \

--disable-libaom --disable-amf --disable-libcodec2 --disable-libdav1d --disable-libdavs2 --disable-libuavs3d --disable-libmfx --disable-libgsm --disable-libkvazaar --disable-libmp3lame --disable-libilbc --disable-libjxl --disable-libvpx --disable-libopencore-amrnb --disable-libopencore-amrwb --disable-libvo-amrwbenc --disable-libfdk-aac --disable-libopenh264 --disable-libopenjpeg --disable-librav1e --disable-libsvtav1 --disable-libtwolame --disable-libx264 --disable-libx265 --disable-libxavs --disable-libxavs2 --disable-encoder=vorbis --disable-libwebp --disable-libopus --disable-libsnappy --disable-libtheora --disable-libxvid

Пишем утилиту работы с медиапотоками

https://github.com/SLenik/dotnext-ffmpeg-muxing-utility/

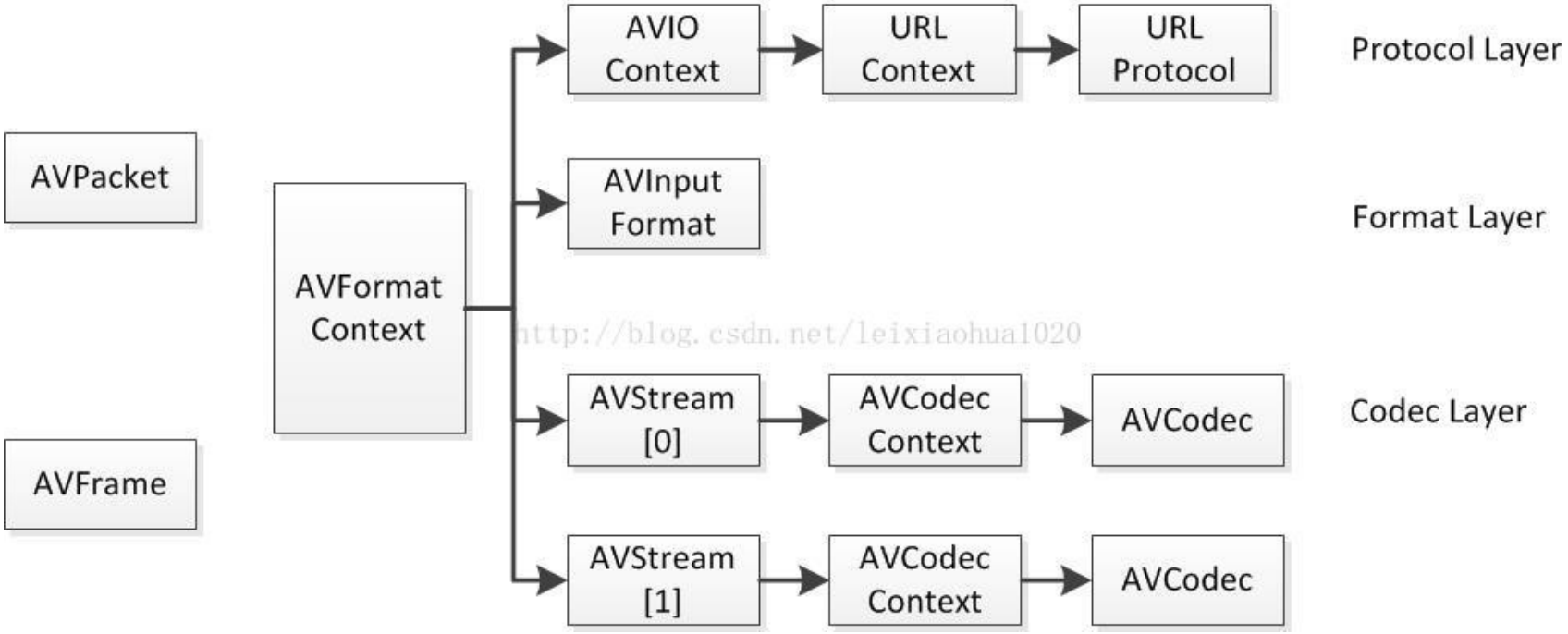
FFmpeg.AutoGen

Кроссплатформенный пакет для вызова методов из библиотек ffmpeg

Как подключить:

- Добавить пакет через nuget.
- Прописать в файле проекта автоматические копирование необходимых библиотек в зависимости от собираемой версии.
- Прописать путь, по которому приложение будет искать эти библиотеки при старте (ffmpeg.RootPath).

Ffmpeg: ключевые структуры



Источник: https://blog.csdn.net/leixiaohua1020/article/details/41181155

Итоги

- Ознакомились с терминологией (контейнер, медиапоток, кодек).
- Разобрались с вариантами получения сборки ffmpeg с нужными нам функциями.
- Реализовали утилиту, осуществляющую ремуксинг файла без перекодирования медиапотоков.

Спасибо за внимание!