

# FFMPEG в приложениях .NET

для работы с медиафайлами

Андреевский 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** (mixing) – объединение нескольких медиапоточков в один контейнер. **Демультимплексирование** (demixing) – разделение контейнера на несколько медиапоточков.

# 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	<a href="#">Полный список доступен в wiki</a> , 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

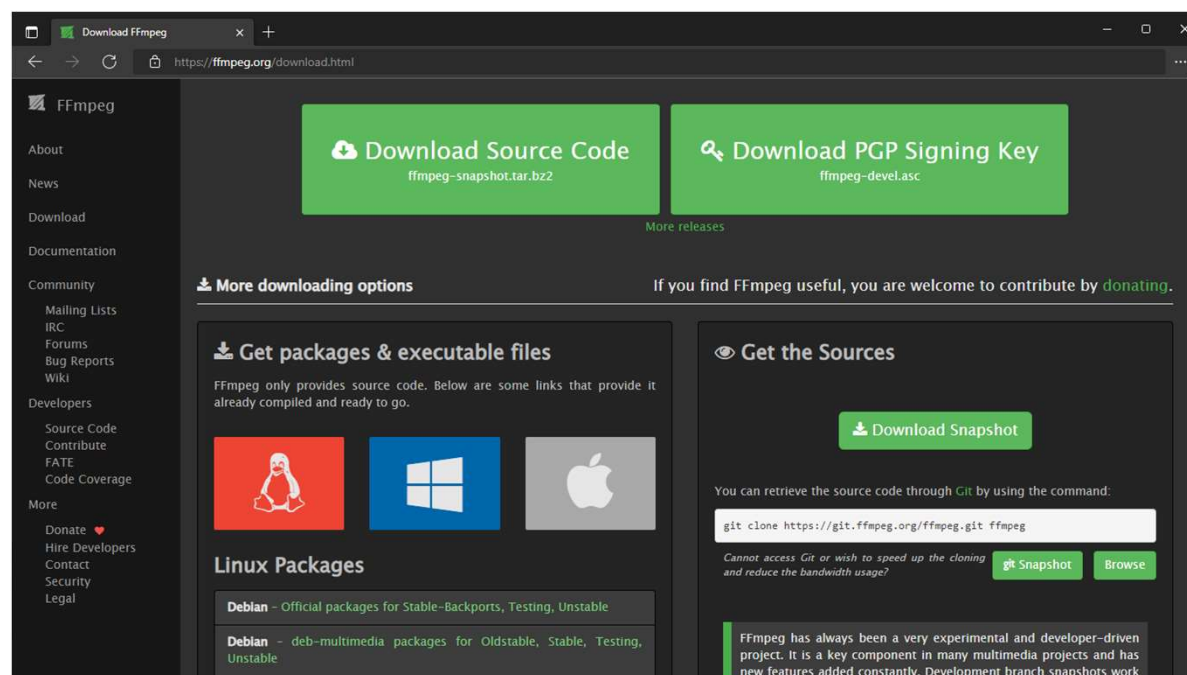
## Через вызовы API библиотек

- 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

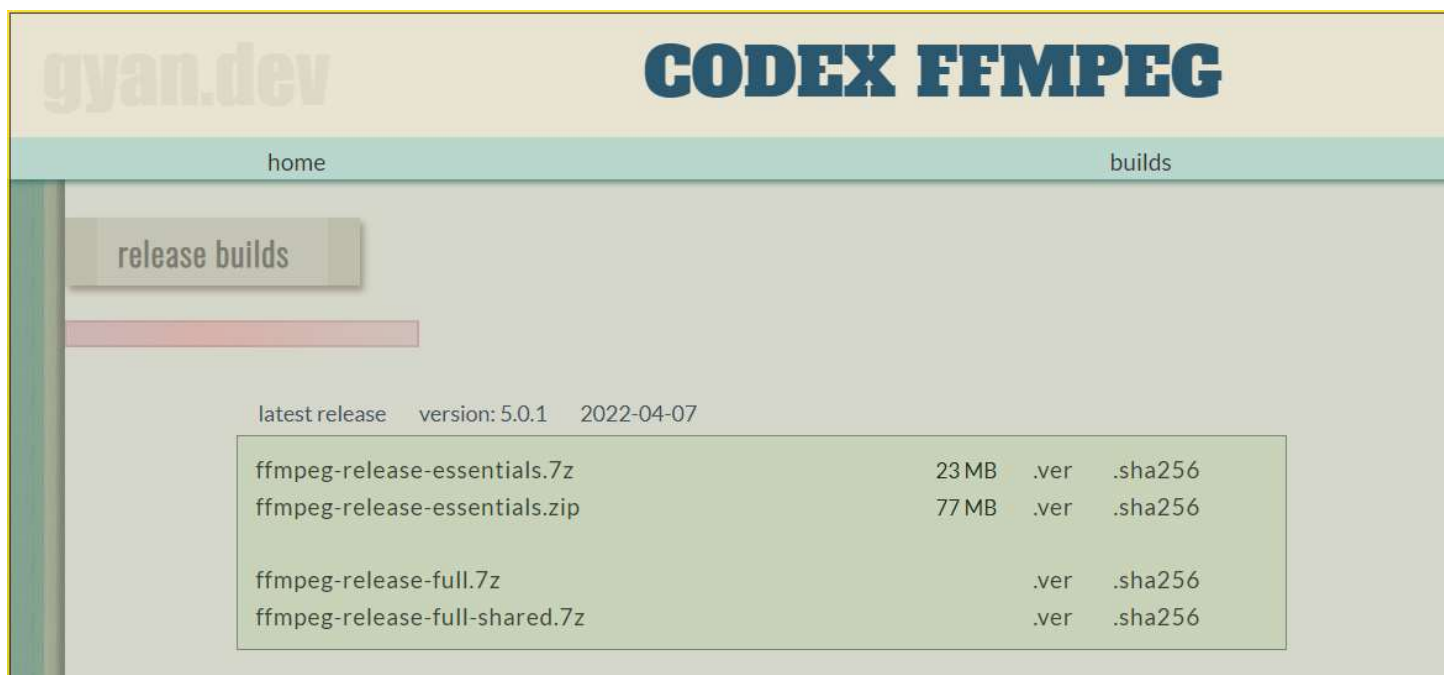
Для себя или для разработки – можно скачать уже скомпилированную

- [Официальная страница ffmpeg](#)
- [ffmpeg-builds на github](#)
- Сборки ffmpeg от Zeranoe (остались доступны лишь через [web.archive.org](http://web.archive.org))



**Обязательно проверяйте список добавленных в сборку библиотек!**

# Сборка FFMPEG: берём готовую

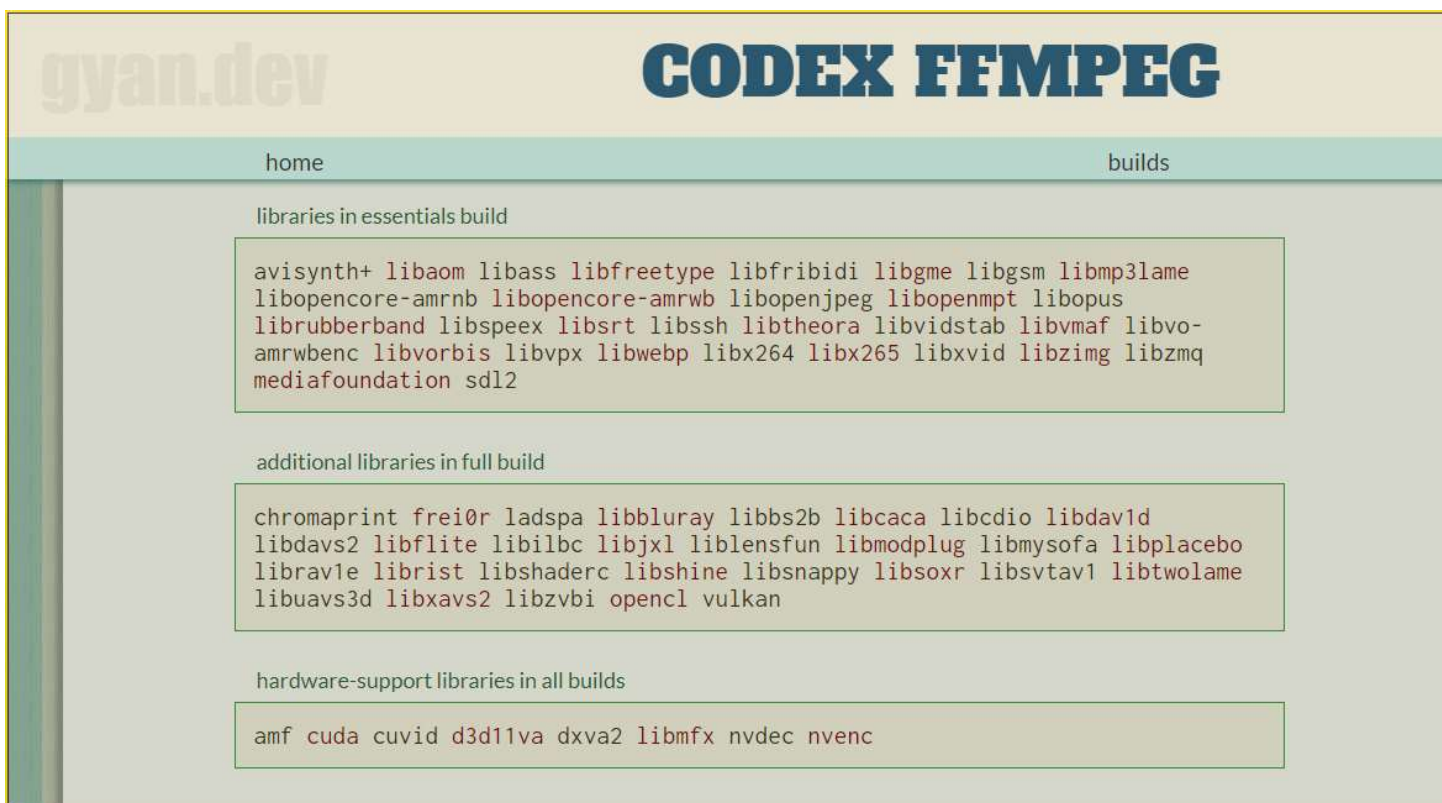


The screenshot shows the website for CODEX FFMPEG. The header includes the logo 'gyan.dev' and the title 'CODEX FFMPEG'. Below the header, there are navigation links for 'home' and 'builds'. A sidebar on the left contains a 'release builds' button. The main content area displays the latest release information: 'latest release version: 5.0.1 2022-04-07'. Below this, a table lists four release builds with their respective sizes and file formats.

latest release version: 5.0.1 2022-04-07			
ffmpeg-release-essentials.7z	23 MB	.ver	.sha256
ffmpeg-release-essentials.zip	77 MB	.ver	.sha256
ffmpeg-release-full.7z		.ver	.sha256
ffmpeg-release-full-shared.7z		.ver	.sha256



# Сборка FFMPEG: берём готовую



The screenshot shows the website 'CODEX FFMPEG' with a navigation bar containing 'home' and 'builds'. The main content area lists libraries for three different build configurations: 'libraries in essentials build', 'additional libraries in full build', and 'hardware-support libraries in all builds'. Each list is enclosed in a light green box.

**gyan.dev** **CODEX FFMPEG**

home builds

libraries in essentials build

```
avisynth+ libaom libass libfreetype libfribidi libgme libgsm libmp3lame  
libopencore-amrnb libopencore-amrwb libopenjpeg libopenmpt libopus  
librubberband libspeex libsrt libssh libtheora libvidstab libvmaf libvo-  
amrwbenc libvorbis libvpx libwebp libx264 libx265 libxvid libzim libzmq  
mediafoundation sdl2
```

additional libraries in full build

```
chromaprint frei0r ladspa libbluray libbs2b libcacca libcdio libdav1d  
libdavs2 libflite libilbc libjxl liblensfun libmodplug libmysofa libplacebo  
librav1e librist libshaderc libshine libsnappy libsoxr libsvtav1 libtwolame  
libuavs3d libxavs2 libzvbi openc1 vulkan
```

hardware-support libraries in all builds

```
amf cuda cuvid d3d11va dxva2 libmfx nvdec nvenc
```

# Сборка FFmpeg: компилируем сами

Если FFmpeg требуется 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-libsnelly --disable-libtheora --disable-libxvid
```

# Пишем утилиту работы с медиапотоками

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

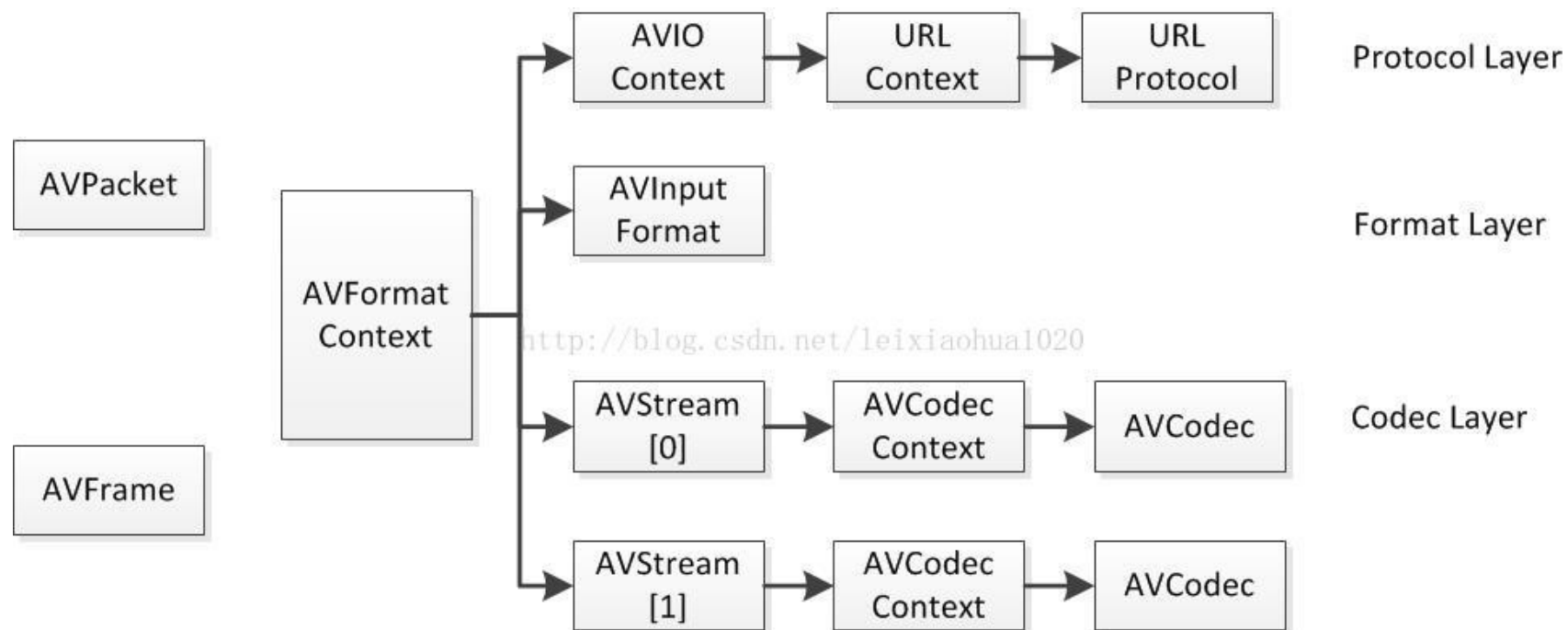
# FFmpeg.AutoGen

Кроссплатформенный пакет для вызова методов из библиотек ffmpeg

Как подключить:

- Добавить пакет через nuget.
- Прописать в файле проекта автоматическое 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 медиапоточков.

**Спасибо за внимание!**