Compile FFmpeg with Hardware Acceleration

FFmpeg is a collection of libraries and tools to process multimedia content such as audio, video, subtitles and related metadata. It is provided as an open-source project, and requires users to compile it on the target machine. When compiling on Raspberry Pi, there are some settings and libraries needed to enable OMX Hardware Acceleration for a faster encoding / decoding.

#pi #ffmpeg #hardware accelerator #omx

Last update: 2021-06-03 14:39:40

Table of Content

Pre-built FFmpeg in Raspberry Pi OS

The FFmpeg package in Raspberry Pi OS is built with H264 Hardware Acceleration already, just need to download it from the package manager:

```
sudo apt install ffmpeg -y
```

The below guide helps to compile FFmpeg from the latest source code to get new features or bug fixes.

1. Manual compilation

Autobuild script

There is a simpler method to compile FFmpeg with an auto-build script. This manual build section is to understand what are needed to be able to compile FFmpeg. It is recommended to use script.

1.1. Update the system

Before starting to build, update system packages to the latest version may help to solve some issues which could happen due to the requirements from the latest source code of FFmpeg.

```
sudo apt update
sudo apt upgrade
```

1.2. Install build package

To compile a source code, system needs to install build tools, including config parsers, compilers, dependent libraries.

Here is the command to install necessary packages:

```
sudo apt -y install autoconf automake build-essential cmake doxygen git graphviz
imagemagick libasound2-dev libass-dev libavcodec-dev libavdevice-dev libavfilter-
dev libavformat-dev libavutil-dev libfreetype6-dev libgmp-dev libmp3lame-dev
libopencore-amrnb-dev libopencore-amrwb-dev libopus-dev librtmp-dev libsdl2-dev
libsdl2-image-dev libsdl2-mixer-dev libsdl2-net-dev libsdl2-ttf-dev libsnappy-dev
libsoxr-dev libssh-dev libssl-dev libtool libv4l-dev libva-dev libvdpau-dev
libvo-amrwbenc-dev libvorbis-dev libwebp-dev libx264-dev libx265-dev libxcb-
shape0-dev libxcb-shm0-dev libxcb-xfixes0-dev libxcb1-dev libxml2-dev lzma-dev
meson nasm pkg-config python3-dev python3-pip texinfo wget yasm zlib1g-dev
libdrm-dev
```

1.3. Compile additional libraries

There are some additional libraries required by FFmpeg, which are not pre-built and distributed in OS package manager.

Create a folder to get started:

```
mkdir ~/ffmpeg-libraries
```

AAC sound format (fdk-aac)

```
git clone --depth 1 https://github.com/mstorsjo/fdk-aac.git ~/ffmpeg-
libraries/fdk-aac \
  && cd ~/ffmpeg-libraries/fdk-aac \
  && autoreconf -fiv \
  && ./configure \
  && make -j$(nproc) \
  && sudo make install
```

AV1 video format (dav1d)

```
git clone --depth 1 https://code.videolan.org/videolan/dav1d.git ~/ffmpeg-
libraries/dav1d \
  && mkdir ~/ffmpeg-libraries/dav1d/build \
  && cd ~/ffmpeg-libraries/dav1d/build \
  && meson .. \
  && ninja \
  && sudo ninja install
```

HEVC encoder (kvazaar)

```
git clone --depth 1 https://github.com/ultravideo/kvazaar.git ~/ffmpeg-
libraries/kvazaar \
  && cd ~/ffmpeg-libraries/kvazaar \
  && ./autogen.sh \
  && ./configure \
  && make -j$(nproc) \
  && sudo make install
```

VP8 and VP9 video codecs (LibVPX)

```
git clone --depth 1 https://chromium.googlesource.com/webm/libvpx ~/ffmpeg-
libraries/libvpx \
  && cd ~/ffmpeg-libraries/libvpx \
  && ./configure --disable-examples --disable-tools --disable-unit_tests --
```

```
disable-docs \
  && make -j$(nproc) \
  && sudo make install
```

AP1 video codec (aom)

```
git clone --depth 1 https://aomedia.googlesource.com/aom ~/ffmpeg-libraries/aom \
  && mkdir ~/ffmpeg-libraries/aom/aom_build \
  && cd ~/ffmpeg-libraries/aom/aom_build \
  && cmake -G "Unix Makefiles" AOM_SRC -DENABLE_NASM=on -
DPYTHON_EXECUTABLE="$(which python3)" -DCMAKE_C_FLAGS="-mfpu=vfp -mfloat-
abi=hard" .. \
  && sed -i 's/ENABLE_NEON:BOOL=ON/ENABLE_NEON:BOOL=OFF/' CMakeCache.txt \
  && make -j$(nproc) \
  && sudo make install
```

Image processing (zimg)

```
git clone -b release-2.9.3 https://github.com/sekrit-twc/zimg.git ~/ffmpeg-
libraries/zimg \
  && cd ~/ffmpeg-libraries/zimg \
  && sh autogen.sh \
  && ./configure \
  && make \
  && sudo make install
```

1.4. Link compiled libraries

After installing new libraries, system needs to refresh link cache for new packages. This command ensures system won't run into linking issues because the compiler can't find a library.

```
sudo ldconfig
```

1.5. Compile FFmpeg

Finally, FFmpeg can be compiled with settings to include additional libraries, and the feature `omx-rpi`. The command is quite large as it has a lot of options:

```
git clone --depth 1 https://github.com/FFmpeg/FFmpeg.git ~/FFmpeg \
  && cd ~/FFmpeg \
  && ./configure \
    --extra-cflags="-I/usr/local/include" \
    --extra-ldflags="-L/usr/local/lib" \
    --extra-libs="-lpthread -lm -latomic" \
    --arch=armel \
    --enable-gmp \
    --enable-gpl \
    --enable-libaom \
    --enable-libass \
    --enable-libdav1d \
    --enable-libdrm \
    --enable-libfdk-aac \
    --enable-libfreetype \
    --enable-libkvazaar \
    --enable-libmp3lame \
    --enable-libopencore-amrnb \
    --enable-libopencore-amrwb \
    --enable-libopus \
    --enable-librtmp \
    --enable-libsnappy \
    --enable-libsoxr \
    --enable-libssh \
    --enable-libvorbis \
    --enable-libvpx \
    --enable-libzimg \
    --enable-libwebp \
    --enable-libx264 \
    --enable-libx265 \
    --enable-libxml2 \
    --enable-mmal \
    --enable-nonfree \
    --enable-omx \
    --enable-omx-rpi \
    --enable-version3 \
    --target-os=linux \
    --enable-pthreads \
    --enable-openssl \
    --enable-hardcoded-tables \
  && make -j$(nproc) \
  && sudo make install
```

The compilation time is quite long, usually 5 hours on Raspberry Pi WiFi Zero, so be patient. Restart the system when the compilation ends, and check for the supported Hardware Acceleration codec:

```
ffmpeg -hide_banner -encoders | grep -E "h264|mjpeg"
```

```
V..... libx264              libx264 H.264 / AVC / MPEG-4 AVC / MPEG-4 part 10 (codec
h264)
V..... libx264rgb           libx264 H.264 / AVC / MPEG-4 AVC / MPEG-4 part 10 RGB
(codec h264)
V..... h264_omx             OpenMAX IL H.264 video encoder (codec h264)
V..... h264_v4l2m2m         V4L2 mem2mem H.264 encoder wrapper (codec h264)
VFS... mjpeg                MJPEG (Motion JPEG)
```

2. Autobuild script

Thank cdgriffith for the below awesome *pi_streaming_setup* script.

2.1. The `pi_streaming_setup` script

There are many guides published on the internet but pi_streaming_setup is a very easy script to follow.

This script is designed to help automate turning a Paspberry Pi with a compatible camera into a MPEG-DASH / HLS streaming server.

The steps it will attempt to do:

1. Install FFmpeg (or compile it before install) with H264 hardware acceleration and free libraries
2. Install NGINX for DASH / HLS or install RTSP server if desired
3. (DASH/HLS) Update *rc.local* to run required setup script on reboot
4. (DASH/HLS) Create *index.html* file to view video stream
5. Create a *systemd* service and enable it to start streaming

> This script requires Python 3.6+

The usage of this script is simple and clear, but to compile FFmpeg, just need 2 options:

```
--compile-ffmpeg
--compile-only
```

2.2. Compile FFmpeg

Install `git` if not installed:

```
sudo apt-get install git
```

Clone the `pi_streaming_setup` repo from github:

```
cd ~
git clone https://github.com/cdgriffith/pi_streaming_setup.git
```

Go into the script's folder:

```
cd pi_streaming_setup
```

and finally, run the script with `sudo` and `python3` as user `pi` :

```
sudo python3 streaming_setup.py --compile-ffmpeg --compile-only --run-as pi
```

This will take about 4~5 hours on an old and slow Raspberry Pi, such as a Pi Zero.

2.3. Test compiled FFmpeg

After the compilation finishes, reboot the Pi, and when it's booted up, run below command to check the compiled tool:

```
ffmpeg -hide_banner -encoders | grep -E "h264|mjpeg"
```

and check the supported codecs:

```
V..... libx264              libx264 H.264 / AVC / MPEG-4 AVC / MPEG-4 part 10 (codec
h264)
V..... libx264rgb           libx264 H.264 / AVC / MPEG-4 AVC / MPEG-4 part 10 RGB
(codec h264)
V..... h264_omx             OpenMAX IL H.264 video encoder (codec h264)
V..... h264_v4l2m2m         V4L2 mem2mem H.264 encoder wrapper (codec h264)
VFS... mjpeg                MJPEG (Motion JPEG)
```