EBook Gratis

APRENDIZAJE

ffmpeg

Free unaffiliated eBook created from
Stack Overflow contributors.

#ffmpeg

Tabla de contenido

Acerca de

You can share this PDF with anyone you feel could benefit from it, downloaded the latest version from: ffmpeg

It is an unofficial and free ffmpeg ebook created for educational purposes. All the content is extracted from Stack Overflow Documentation, which is written by many hardworking individuals at Stack Overflow. It is neither affiliated with Stack Overflow nor official ffmpeg.

Use the content presented in this book at your own risk; it is not guaranteed to be correct nor accurate, please send your feedback and corrections to info@zzzprojects.com

Capítulo 1: Empezando con ffmpeg

Observaciones

FFMpeg Esta sección proporciona una descripción general de qué es ffmpeg y por qué un desarrollador puede querer usarlo.

También debe mencionar cualquier tema grande dentro de ffmpeg, y vincular a los temas relacionados. Dado que la Documentación para ffmpeg es nueva, es posible que deba crear versiones iniciales de esos temas relacionados.

Examples

Instalación o configuración

FFmpeg se puede instalar en una combinación de sistemas operativos, incluidos Unix y OS X. Con una extensión de línea de comandos, esta utilidad también se puede instalar en Windows con una DLL.

OS X

Para instalar esta utilidad en OS X, solo diríjase a ffmpeg.org , descargue la versión relativa a la arquitectura de su Mac (las instrucciones para encontrar esto se pueden encontrar aquí). Luego coloque la aplicación en un directorio accesible y ejecútela desde la línea de comandos.

Otra forma es usar HomeBrew: https://www.howtogeek.com/211541/homebrew-for-os-x-easily-installs-desktop-apps-and-terminal-utilities/

Por ejemplo

```
brew install ffmpeg --with-fdk-aac --with-ffplay --with-libass --with-libvorbis --with-libvpx
--with-rtmpdump --with-openh264 --with-tools
```

Windows

Para instalar esta utilidad en Windows, diríjase a [ffmpeg.org] (https://www.ffmpeg.org/download.html#build-windows) y siga el enlace de descarga, usando su arquitectura. Las instrucciones para encontrar esto se pueden ver [aquí] (http://answers.microsoft.com/en-us/windows/forum/windows_7-hardware/i-need-to-know-how-to-determine-my-processors / 3ede9c69-25f5-427b-8e8d-e9dd2d032d22). Luego coloque el software descargado en un directorio accesible y ejecútelo desde la línea de comandos.

Unix

Para instalar esta utilidad en Unix, simplemente siga las instrucciones que se encuentran en [ffmpeg.org] (https://www.ffmpeg.org/download.html#build-linux)

Para verificar si ffmpeg está instalado correctamente y ver una lista de los comandos disponibles, intente ejecutar el siguiente comando en la línea de comandos:

```
ffmpeg -help
```

¿Qué es FFmpeg?

FFmpeg (o "Fast Forward MPEG") es una utilidad de línea de comandos simple pero con muchas funciones para permitir la manipulación (incluida la decodificación, codificación, transcodificación, mezcla, demuxación, transmisión, filtrado y reproducción) de archivos multimedia y de video. Esta utilidad difiere de otro software orientado a GUI, ya que emplea la metodología WYSIWYG (Lo que ves es lo que obtienes). En lugar de los menús y funciones ocultos, todo se puede encontrar simplemente escribiendo `ffmpeg -h` cuando se configura, o siguiendo la documentación completa . Además de la herramienta de línea de comandos, hay una serie de bibliotecas de C (que utiliza), que se pueden usar para llevar la funcionalidad de FFmpeg a otros proyectos. La documentación de las bibliotecas incluye muchos ejemplos para ayudarlo a comenzar.

Lea Empezando con ffmpeg en línea: https://riptutorial.com/es/ffmpeg/topic/4935/empezando-con-ffmpeg

Capítulo 2: Descodificación

Introducción

Obtención del video / audio en bruto de flujos de medios codificados.

Examples

Encontrar una corriente

Los contenedores de transmisión de medios generalmente tienen varias transmisiones, como una transmisión de video y una transmisión de audio. Por ejemplo, puede obtener la transmisión de audio utilizando lo siguiente:

```
// A Format Context - see Reading Data for more info
AVFormatContext *formatContext;

// Inspect packets of stream to determine properties
if (avformat_find_stream_info(formatContext, NULL) < 0){
    // Error finding info
}

// Find the stream and its codec
AVCodec* audioCodec;
int audioStreamIndex = av_find_best_stream(
    formatContext,        // The media stream
    AVMEDIA_TYPE_AUDIO,   // The type of stream we are looking for - audio for example
    -1,                   // Desired stream number, -1 for any
    -1,                   // Number of related stream, -1 for none
    &audioCodec,          // Gets the codec associated with the stream, can be NULL
    0                     // Flags - not used currently
);
if(audioStreamIndex = AVERROR_STREAM_NOT_FOUND || !audioCodec){
    // Error finding audio (ie. no audio stream?)
}
```

Para obtener otros tipos de flujos, solo necesita reemplazar el tipo de flujo. Los siguientes son tipos válidos:

```
AVMEDIA_TYPE_VIDEO,
AVMEDIA_TYPE_AUDIO,
AVMEDIA_TYPE_SUBTITLE,
AVMEDIA_TYPE_DATA,        // Usually continuous
AVMEDIA_TYPE_ATTACHMENT,  // Usually sparse
```

Abrir un contexto de codec

Una vez que tenga un Contexto de Formato de transmisión y su Códec respectivo, puede abrirlo para decodificar usando el siguiente código:

```
// The format context and codec, given - see Find a stream for how to get these
AVFormatContext *formatContext;
AVCodec* codec;
int streamIndex;

// Get the codec context
AVCodecContext *codecContext = avcodec_alloc_context3(codec);
if (!codecContext){
    // Out of memory
    avformat_close_input(&formatContext);
}

// Set the parameters of the codec context from the stream
int result = avcodec_parameters_to_context(
    codecContext,
    formatContext->streams[streamIndex]->codecpar
);
if(result < 0){
    // Failed to set parameters
    avformat_close_input(&formatContext);
    avcodec_free_context(&codecContext);
}

// Ready to open stream based on previous parameters
// Third parameter (NULL) is optional dictionary settings
if (avcodec_open2(codecContext, codec, NULL) < 0){
    // Cannot open the video codec
    codecContext = nullptr;
}

// Do something with the opened codec context... (ie decode frames through the context)
```

Decodificar marcos

Dado un contexto de códec y paquetes codificados de un flujo de medios, puede comenzar a decodificar medios en cuadros sin procesar. Para decodificar un solo cuadro, puedes usar el siguiente código:

```
// A codec context, and some encoded data packet from a stream/file, given.
AVCodecContext *codecContext;  // See Open a codec context
AVPacket *packet;              // See the Reading Media topic


// Send the data packet to the decoder
int sendPacketResult = avcodec_send_packet(codecContext, packet);
if (sendPacketResult == AVERROR(EAGAIN)){
    // Decoder can't take packets right now. Make sure you are draining it.
}else if (sendPacketResult < 0){
    // Failed to send the packet to the decoder
}

// Get decoded frame from decoder
AVFrame *frame = av_frame_alloc();
int decodeFrame = avcodec_receive_frame(codecContext, frame);

if (decodeFrame == AVERROR(EAGAIN)){
    // The decoder doesn't have enough data to produce a frame
    // Not an error unless we reached the end of the stream
```

```
    // Just pass more packets until it has enough to produce a frame
    av_frame_unref(frame);
    av_freep(frame);
}else if (decodeFrame < 0){
    // Failed to get a frame from the decoder
    av_frame_unref(frame);
    av_freep(frame);
}

// Use the frame (ie. display it)
```

Si desea descodificar *todos los* marcos, puede simplemente colocar el código anterior en un bucle, alimentándolo con paquetes consecutivos.

Lea Descodificación en línea: https://riptutorial.com/es/ffmpeg/topic/10090/descodificacion

Capítulo 3: Ffmpeg Restream

Examples

Dispositivo simple Restream

Ffmpeg es un cuchillo suizo para proyectos de streaming. Para cualquier tipo de dispositivo de transmisión solo necesita obtener la especificación del dispositivo. Para enumerar el dispositivo

```
ffmpeg -f dshow -list_devices true  -i dummy
```

La línea de comandos mostrará una lista de todos los dispositivos disponibles en la máquina.

```
[dshow @ 0000000004052420] DirectShow video devices
[dshow @ 0000000004052420]  "ManyCam Virtual Webcam"
[dshow @ 0000000004052420]  "UScreenCapture"
[dshow @ 0000000004052420] DirectShow audio devices
```

Para reiniciar el dispositivo de audio y video,

```
ffmpeg -f dshow -i video="DirectShow video devices":audio="DirectShow audio devices"
-vcodec libx264 -acodec aac -strict experimental 2 -tune zerolatency -f flv
rmtp://WOWZA_IP/WOWZA_APP/STREAMNAME
```

Esto puede extenderse a todo tipo de dispositivos como dispositivos médicos o hardware de video.

Lea Ffmpeg Restream en línea: https://riptutorial.com/es/ffmpeg/topic/9517/ffmpeg-restream

Capítulo 4: Medios de lectura

Introducción

Hay algunas maneras de leer Audio / Video en FFmpeg.

Examples

Leyendo de memoria

libavformat usualmente toma un nombre de archivo y lee los medios directamente desde el sistema de archivos. Si desea leer de la memoria (como las secuencias), haga lo siguiente:

```
// Define your buffer size
const int FILESTREAMBUFFERSZ = 8192;


// A IStream - you choose where it comes from
IStream* fileStreamData;

// Alloc a buffer for the stream
unsigned char* fileStreamBuffer = (unsigned char*)av_malloc(FILESTREAMBUFFERSZ);
if (fileStreamBuffer == nullptr){
    // out of memory
}

// Get a AVContext stream
AVIOContext* ioContext = avio_alloc_context(
    fileStreamBuffer,    // Buffer
    FILESTREAMBUFFERSZ,  // Buffer size
    0,                   // Buffer is only readable - set to 1 for read/write
    fileStreamData,      // User (your) specified data
    FileStreamRead,      // Function - Reading Packets (see example)
    0,                   // Function - Write Packets
    FileStreamSeek       // Function - Seek to position in stream (see example)
);
if(ioContext == nullptr){
    // out of memory
}

// Allocate a AVContext
AVFormatContext *formatContext = avformat_alloc_context();

// Set up the Format Context
formatContext->pb = ioContext;
formatContext->flags |= AVFMT_FLAG_CUSTOM_IO; // we set up our own IO

// Open "file" (open our custom IO)
// Empty string is where filename would go. Doesn't matter since we aren't reading a file
// NULL params are format and demuxer settings, respectively
if (avformat_open_input(&formatContext, "", nullptr, nullptr) < 0){
    // Error opening file
}
```

```
// Do something with the formatContext

// Free resources!
avformat_close_input(&formatContext);
av_free(ioContext);
```

Leyendo de un archivo

Abrir un archivo multimedia desde el sistema de archivos local.

```
AVFormatContext *formatContext;

// Open the file
if(avformat_open_file(&formatContext, "path/to/file.ogg", NULL, NULL) < 0){
    // Error opening file
}

// Do something with the file

// Free resources
avformat_close_input(&formatContext);
```

Leyendo desde un contexto de formato

Los formatos contienen uno o más flujos codificados y multiplexados. Por lo general, los leemos en fragmentos, que a menudo se denominan cuadros (aunque en ciertos casos, FFmpeg se refiere exclusivamente a fragmentos de medios sin formato descodificados como cuadros, y fragmentos codificados como paquetes, que pueden ser confusos). Para leer un solo cuadro desde un formato, use lo siguiente:

```
// A Format Context - see other examples on how to create it
AVFormatContext *formatContext;

// Initialize the AVPacket manually
AVPacket avPacket;
av_init_packet(&avPacket); // set fields of avPacket to default.
avPacket.data = NULL;
avPacket.size = 0;

// Read from the context into the packet
if(av_read_frame(formatContext, &avPacket) == 0){
    // nothing read
}

// Use the packet (such as decoding it and playing it)

// Free packet
av_packet_unref(&avPacket);
```

Leyendo de un IStream en un IOContext

La llamada a la API `avio_alloc_context` , que configura un contexto IO personalizado, toma un

puntero a una función de lectura. Si está leyendo de un IStream, puede usar lo siguiente:

```
/**
 * Reads from an IStream into FFmpeg.
 *
 * @param ptr       A pointer to the user-defined IO data structure.
 * @param buf       A buffer to read into.
 * @param buf_size  The size of the buffer buff.
 *
 * @return The number of bytes read into the buffer.
 */
int FileStreamRead(void* ptr, uint8_t* buf, int buf_size)
{
    // This is your IStream
    IStream* stream = reinterpret_cast<IStream*>(ptr);

    ULONG bytesRead = 0;
    HRESULT hr = stream->Read(buf, buf_size, &bytesRead);
    if(hr == S_FALSE)
        return AVERROR_EOF;  // End of file

    if(FAILED(hr))
        return -1;
    return bytesRead;
}
```

Buscando dentro de un IStream en un IOContext

La llamada a la API `avio_alloc_context` , que configura un contexto IO personalizado, toma un puntero a una función de **Búsqueda** . Si está leyendo de un IStream, puede usar lo siguiente:

```
/**
 * Seeks to a given position on an IStream.
 *
 * @param ptr     A pointer to the user-defined IO data structure.
 * @param pos     The position to seek to.
 * @param origin  The relative point (origin) from which the seek is performed.
 *
 * @return  The new position in the IStream.
 */
int64_t FileStreamSeek(void* ptr, int64_t pos, int origin){
    // Your custom IStream
    IStream* stream = reinterpret_cast<IStream*>(ptr);

    // Prevent overflows
    LARGE_INTEGER in = { pos };
    ULARGE_INTEGER out = { 0 };

    // Origin is an item from STREAM_SEEK enum.
    //   STREAM_SEEK_SET - relative to beginning of stream.
    //   STREAM_SEEK_CUR - relative to current position in stream.
    //   STREAM_SEEK_END - relative to end of stream.
    if(FAILED(stream->Seek(in, origin, &out)))
         return -1;

    // Return the new position
    return out.QuadPart;
}
```

Lea Medios de lectura en línea: https://riptutorial.com/es/ffmpeg/topic/10089/medios-de-lectura

Creditos

S. No	Capítulos	Contributors
1	Empezando con ffmpeg	Abex, Chris, Community, drumkruk, Olga Khylkouskaya, RhysO
2	Descodificación	JCOC611
3	Ffmpeg Restream	Emre Karataşoğlu
4	Medios de lectura	JCOC611