無料電子ブック

学習

ffmpeg

Free unaffiliated eBook created from
Stack Overflow contributors.

#ffmpeg

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

1: ffmpegをいめる

FFmpegこのセクションでは、ffmpegのとがなぜそれをいたいのかをします。

また、ffmpegのきなテーマについてもし、するトピックにリンクするがあります。 ffmpegのドキュメンテーションはしいので、これらのトピックのバージョンをするがあります。

Examples

インストールまたはセットアップ

FFmpegは、UnixやOS Xをむオペレーティングシステムのにインストールすることができます。コマンドラインをして、このユーティリティはdllをしてWindowsにインストールすることもできます。

OS X

このユーティリティをOS Xにインストールするには、 ffmpeg.orgにかい 、あなたのMacsアーキテクチャにしたリリースをダウンロードしてくださいこれについては、 ここでつけることができます 。に、アプリケーションをアクセスなディレクトリにき、コマンドラインからします。

のは、HomeBrewをすることです https ://www.howtogeek.com/211541/homebrew-for-os-x-easily-installs-desktop-apps-and-terminal-utilities/

えば

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

Windows

このユーティリティをWindowsにインストールするには、[ffmpeg.org] https://www.ffmpeg.org/download.html#build-windowsにし、アーキテクチャをしてダウンロードリンクにってください。これをつけるはこちら[ここ]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。に、ダウンロードしたソフトウェアをアクセスなディレクトリにき、コマンドラインからします。

Unix

このユーティリティをUnixにインストールするには、[ffmpeg.org] https://www.ffmpeg.org/download.html#build-linuxのにってください。

ffmpegがしくインストールされているかどうかをし、なコマンドのをるには、コマンドラインでのコマンドをしてみてください。

```
ffmpeg -help
```

FFmpegとはですか

FFmpeg または "Fast Forward MPEG"は、メディアやビデオファイルのデコード、エンコーディング、トランスコード、、、ストリーミング、フィルタリング、などをにするシンプルでなコマンドラインユーティリティです。このユーティリティは、WYSIWYGメソドロジあなたがているものはですかをしているため、のGUIソフトウェアとはなります。されたメニューやのわりに、セットアップに`ffmpeg -h`とするか、 なにうだけですべてがつかります 。コマンドラインツールにえて、FFmpegのをのプロジェクトにもたらすためにできるいくつかのCライブラリそれがするがあります。のには、めるためのくのがあります。

オンラインでffmpegをいめるをむ https://riptutorial.com/ja/ffmpeg/topic/4935/ffmpegをいめる

2: Ffmpeg Restream

Examples

シンプルなデバイスリストリーム

Ffmpegはストリーミングプロジェクトのスイスナイフです。どんなのデバイスストリーミングでも、デバイスのをするだけでみます。デバイスをするには

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

コマンドプロンプトには、マシンのなすべてのデバイスがされます。

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

オーディオおよびビデオデバイスをするために、

```
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
```

これは、やビデオハードウェアのようなあらゆるのにすることができます。

オンラインでFfmpeg Restreamをむ https://riptutorial.com/ja/ffmpeg/topic/9517/ffmpeg-restream

3: デコード

き

エンコードされたメディアストリームからのビデオ/オーディオをする。

Examples

ストリームをする

メディアストリームコンテナには、、ビデオストリームやオーディオストリームなどののストリームがあります。たとえば、のようにしてオーディオストリームをできます。

```
// 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?)
}
```

のタイプのストリームをするには、ストリームのタイプをきえるだけです。なタイプはのとおりです。

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

コーデックコンテキストをく

ストリームフォーマットコンテキストとそれぞれのコーデックをしたら、のコードをしてデコードにくことができます。

```
// 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)
```

デコードフレーム

コーデックコンテキストとメディアストリームからエンコードされたパケットがえられた、メディアののフレームへのデコードをできます。 1つのフレームをデコードするには、のコードをします。

```
// 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)
```

すべてのフレームをデコードするは、のコードをループにき、したパケットをります。

オンラインでデコードをむ https://riptutorial.com/ja/ffmpeg/topic/10090/デコード

4: メディア

き

Audio / VideoをFFmpegにみむはいくつかあります。

Examples

メモリからのみみ

libavformatは、ファイルをりみ、ファイルシステムからメディアをみみます。メモリストリームなどからみるは、のをいます。

```
// 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);
```

ファイルからのみみ

ロ―カルファイルシステムからメディアファイルをく。

```
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);
```

コンテキストからのみみ

には、1つまたはのエンコ―ドされたストリ―ムとされたストリ―ムがまれます。はフレ―ムとばれるチャンクでみみますFFmpegは、デコ―ドされたのメディアチャンクをフレ―ムとして、エンコ―ドされたチャンクをパケットとしてしますが、するがあります。フォ―マットから1つのフレ―ムをみむには、をします。

```
// 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);
```

IOContextのIStreamからのみみ

カスタムIOコンテキストをするAPIびし`avio_alloc_context`は、Readへのポインタをけります。IStreamからみんでいるは、をできます。

```
/**
 * 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;
}
```

IOContextのIStreamでする

カスタムIOコンテキストをするAPIコール`avio_alloc_context`は、 **Seek**へのポインタをけります。IStreamからみんでいるは、をできます。

```
/**
 * 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;
}
```

オンラインでメディアをむ https://riptutorial.com/ja/ffmpeg/topic/10089/メディア

クレジット

S. No		Contributors
1	ffmpegをいめる	Abex, Chris, Community, drumkruk, Olga Khylkouskaya, RhysO
2	Ffmpeg Restream	Emre Karataşoğlu
3	デコード	JCOC611
4	メディア	JCOC611