Xin Fang
CINE-GT 1807 Digital Preservation
Nicole Martin
11/30/2016
Final Project

FFmpeg

FFmpeg is free software project that serves as the leading multimedia framework to be able to encode, decode and transcode various audio/visual formats.[1] FFmpeg has utilitiy libraries including libavcodec, libavutil[2], libavformat, libavfilter, libavdevice[3], libswscale[4] and libswresample[5], which can be used by developers for applications, as well as four command line tools including ffmpeg, ffserver, ffplay and ffprobe, which can be used by end users for manipulating audio/visual files. This paper mostly focuses on ffmpeg the command line tool and how it is used in the environment of digital preservation.

Considering the nature of FFmpeg project as being free software, which means users can run the software for any purpose as well as to study, it is widely used in different environments. FFmpeg is published under the GNU Lesser General Public License 2.1+ or GNU General Public License 2+ (depends on which parts of FFmpeg project one is referring to).[6] Both of the licenses allow end users to have the freedom to run, study, share and modify the software. Due to the nature of being a free software project, there are several open forums, mailing lists and website hubs for users to develop and discuss FFmepg.

---

1 "About FFmpeg". FFmpeg.. Web. Dec. 9, 2016. < https://ffmpeg.org/about.html>.
2 The utility library to aid portable multimedia programming.
3 The library that includes input and output device.
4 For image scaling and colorspace and pixel format conversion.
5 For audio resampling, rematrixing and format conversion.
6 "FFmepg" Wikipedia: The Free Encyclopedia. Wikimedia Foundation, Inc. 12 Dec. 2016. Web. 12 Dec. 2016 < https://en.wikipedia.org/wiki/FFmpeg>

ffmpeg: fundamentals

The ffmpeg command line tool could be complied in different platform including Linux, Mac OS, Microsoft Windows, and even Android on smart phone. ffmpeg is used as audio/visual files converter that could also work on live sources.[7] The process of transcoding the files including demuxing the input file into encoded data packet, and decoding the packet into decoded frames. The decoded frames are then encoded into encoded data packets, following by being muxed into output files. One alternative process is that ffmpeg can process raw audio and video frames using filters from the libavfilter library before encoding.

First, ffmpeg calls the libavformat library, which contains the demuxers, to read the input files. "Demux" is an abbreviated way of saying "demultiplexer"[8]. A demuxer is a module responsible for extracting the contents of a given file[9] format, and it takes one single input data line and then switches it to any one of individual output lines one at a time.[10] When a tool like ffmpeg reads a file, it usually tries to auto detect the file format to select the right demuxer to use. Demuxers are configured elements in FFmpeg that can read the multimedia streams[11] from one certain type of file. When users configure the whole FFmpeg build, all the supported demuxers are enabled by default.

Once the encoded data packets are created through the demuxer, they are then passed into the decoder to be the decoded frames. A decoder decompresses the data into

[7] "ffmpeg Documentation". FFmpeg. Web. Dec. 9, 2016. < https://ffmpeg.org/about.html>
[8] "The Demultiplexer". Electronics Tutorials. Web. Dec. 12. < http://www.electronics-tutorials.ws/combination/comb_3.html>
[9] The word "file" in FFmpeg including not only a digital file but could be pipe, network stream, grabbing devide, etc.
[10] Ibid.
[11] Those words what are underlined in the paper are the terms specifically used in FFmpeg documentations.

uncompressed form such as raw video or PCM audio. Those uncompressed forms could be further processed by a step called filtering. There are three types of filters regards to which part they function in the whole converting process.[12] The one used before encoding is called prefilter, including those filters for denoising[13], deinterlacing[14], deflicking[15] videos. The intrafilter that functions with the video encoder are usually integrated as a part of codec. Postfilters are used after encoding such as deblocking filters[16] and deinterlacing.

Filtering in FFmpeg is enabled through the libavfilter library. A filter can process multiple inputs and outputs. A filter chain consists of a sequence of connected filters in linear order. A filter graph is the visualize version of a whole "system" of connected filter chains.[17] ffmpeg has two types of filter graphs: simple and complex. Filter graph may contain several filter chains, each of which may contain several filters. Simple filter graph includes exactly one input and output, and both of input and output are the same type. The simple filter graph looks like a linear chain, and it could be inserting an additional step between decoding and encoding. The complex filter graph has more than one input or output, or when output type is different from input. ~~A filter without an input is called a "source", and a filter without an output is called a "sink".~~[18]

There are several common filters that has been used very often, including the scale filter and yadif filter. Scale filter is for resizing the video, converting the image format and colors

[12] "Filter(Video)" Wikipedia: The Free Encyclopedia. Wikimedia Foundation, Inc. May 28, 2016. Web. Dec. 9, 2016. < https://en.wikipedia.org/wiki/Filter_(video)>
[13] Video denoising is the process of removing noise from a video signal.
[14] Deinterlacing is the process of converting interlaced video into a non-interlaced form.
[15] Deflicking process applies to brightness flicker in video to improve visual quality.
[16] A part of the specification for H.264 codec.
[17] "FFmpeg Filters Documentation" FFmpeg. Web. Dec. 9, 2016. < https://ffmpeg.org/ffmpeg-filters.html#Filtergraph-description>
[18] I think this is an important knowledge but it is not necessary related to anything I have in this paper.

pace of the image. In ffmpeg, the scale filter is performed through libswscale library. The scale filter forces the output display aspect ratio to be the same as the input one, by changing the output sample aspect ratio.[19] Users need to remember to set the output video dimension expression by using "w" (width) and "h" (height) in the command. Otherwise the default output value is the input dimension. Scale filter is widely used in converting videos. Libavfilter, which is FFmpeg's filter library, will automatically insert scale filters where format conversion is required. yadif filter is for deinterlace the input video. The name "yadif" means "yet another deinterlacing filter". yadif accepts the parameters of choosing which interlacing mode to adopt, choosing between the picture field parity[20], specifying which frames to deinterlace.[21]

The decoded frame goes through the encoder and turns to be the encoded data packet. Then the muxer will transfer it to the output file. Just functioning as the opposite of demuxer, muxer combines all the data packets together into a file. Muxer and demuxer attach to file container format. A codec is the code for both encoding and decoding. Technically, simple filter graph is a step inserting between encoding and decoding, but sometimes people refer to it as decoding because some filters integrated with codec. Filter sometime is referred to as decoder, because their functions are similar. However, filter and codec are two separate concepts.

---

[19] "FFmpeg Filters Documentation. 9.125 Scale" FFmpeg. Web. Dec. 9, 2016. <https://ffmpeg.org/ffmpeg-filters.html#scale-1>
"FFmpeg Scaler Documentation 2. Scaler Options" FFmpeg. Web. Dec. 9, 2016. <https://ffmpeg.org/ffmpeg-scaler.html#scaler_005foptions>

[20] Interlaced video, first and second field.

[21] "FFmpeg Filters Documentation 9.166 yadif" FFmpeg. Web. Dec. 9, 2016. <https://ffmpeg.org/ffmpeg-filters.html#yadif-1>

ffmpeg: basic commands

"ffmpeg [global_options] {[input_file_options] -i input_file} ... {[output_file_options] output_file} ..."[22]

This is the basic syntax for ffmpeg commands. In command line interface, there are several global options such as "--help" or "-h" for displaying detailed help information, and "--version" or "-V" for displaying this application version. Those global options work for every command line tool. The global options in ffmpeg are the ones that affect the whole program instead of just one file. The useful common global options in ffmpeg are "-y" for overwriting the output files and "-n" for never overwriting output files.

"-i" option is required in the command because it follows by the input file. The "file" here could be the regular digital file or network stream. ffmpeg is also able to process multiply input files. Each of the files could contain multiple streams like audio, video, subtitles or data. Depend on which file container format the file has, the number and the type of stream could be limited.

All the options that are mentioned above all are the generic options. There is also what is called AVOptions in ffmpeg. The "input file options" part and "output file options" part are the areas for AVOptions, which are provided directly by the libavformat, libavdevice and libavcodec libraries. The AVOptions have generic and private categories. The generic options

[22] "ffmpeg Documentation 1 Synopsis" FFmpeg. Web. Dec. 9, 2016. <https://ffmpeg.org/ffmpeg.html#Synopsis>

apply for all the containers and private options are for specific containers.[23]

The other important options that are useful such as "-f", which means "force format". It could be used before both input and output files. The file format is normally auto detected by ffmpeg from input files and guessed from the file extension of output files. "-metadata" option allows ffmpeg to change the metadata field for the file. The certain metadata field should be specified by the "key=value" option.[24] The advanced option for metadata is the metadata specifier, which has the syntax of "-map_metadata[:metadata_spec_out] infile[:metadata_spec_in]".[25] Notice that the infile here is for zero-based indices instead of the file name.

The filter and filter graph in ffmpeg take an important part in the section above. The syntax related to the application of filters through ffmpeg should be mentioned here. Filters in linear chain are separated by commas ",". The filter chains are separated by a semicolon ";".[26] If the input or output is not specified, it is assumed to come from the preceding filter or sent to the following item in the chain by default. "-vf" option stands for video filter, and "-af" stands for audio filter. Options including "-filter", "-vf" and "-af" represent a filter graph.[27] In commands, filter names are followed by a string of numbers, which are the parameters for this filter. In ffmpeg, this string of numbers is called arguments. The numbers are separated by colon ":", and there are different options depends on which filter is using.

---

[23] "ffmpeg Documentation 5.3 AVOptions" FFmpeg. Web. Dec. 9, 2016. <https://ffmpeg.org/ffmpeg.html#AVOptions>
[24] "ffmpeg Documentation 5.4 Main options" FFmpeg. Web. Dec. 9, 2016. <https://ffmpeg.org/ffmpeg.html#Main-options>
[25] "ffmpeg Documentation 5.11 Advanced options" FFmpeg, Web. Dec 9, 2016. <https://ffmpeg.org/ffmpeg.html#Advanced-options>
[26] "FFmpeg Filtering Guide" FFmpeg Wiki. Web. Dec. 9, 2016. <https://trac.ffmpeg.org/wiki/FilteringGuide>
[27] Ibid.

ffmpeg: features

ffmpeg has many defaults if users do not specify some options in commands. The most obvious one among those defaults is the stream selection. ffmpeg includes only one stream of video, audio and subtitle track in the input files and adds them to each output file. If users have one ffmpeg command that include several streams in the input file part, ffmpeg would only choose one stream of video, audio and subtitle to perform the change. It picks the stream of the highest quality within each type. For video, the highest quality one is the stream with the highest resolution; for audio, it is the stream with the most channels; for subtitles, it is the first subtitle stream. If there are several streams of the same bitrate, the first stream in command will be chosen. "-map" is the advanced option in ffmpeg to avoid this situation and disable the stream selection default. This option specifies one or more input streams as a source for the output file, and it disables designated streams from already created mappings by default. The "-map_metadata" option I mentioned in the previous section functions in the similar way for metadata. ffmpeg recognize by default that global metadata is copied from the first input file, and the metadata for each stream or chapter is copied along with streams/chapters. "-map" is an advanced, full manual controlled option in ffmpeg. It could be overwhelming to use. Options like "-vn" and "-an", which simply disable video and audio recording, would be easier to understand and control.

On the other hand, it is very time consuming to have ffmpeg perform command on every

stream in the input file, especially when users have preservation master level high quality files. The stream selection default helps to solve this problem for average users. In this case, the stream copy option ("-copy") would be useful.[28] This option helps for ffmpeg to work on all the streams in the input file(s) without quality loss. The stream copy mode is under the "-codec" option. There is no decoding or encoding for the streams that are copied, so those streams do not need to go through encoder, decoder and filters. When "-copy" is added before output file, it selects encoder. When it is used before input file, it selects decoder. Theoretically, it is useful for changing the container format or modifying container-level metadata. Stream copy and disabling stream selection default are commonly used together.

ffmpeg: digital preservation

ffmpeg is mostly used for converting audio/visual files for the daily use basis. Digital preservation is a formal commitment in library science to make sure the digital information is accessible, usable and sustainable. Preserving audio/visual files is a challenge, and incorporating ffmpeg into workflow is a great help. Although ffmpeg could be implemented in different platforms, using bash script in Mac OS is the most ideal way of working with ffmpeg for digital preservation. This section has the suggested usages of ffmpeg based in Mac OS platform.

In digital preservation environment, it is common to process batch of large files. ffmpeg

[28] "ffmpeg Documentation 3.2 Stream copy" FFmpeg. Web. Dec.9, 2016. <https://ffmpeg.org/ffmpeg.html#Stream-copy>

can work on multiple files at the same time. To achieve this, bash script is necessary.

```
for %%a in ("*.*") do ffmpeg -i "%%a" ...
```

In bash script, the syntax “for %%a in ("*.*")” starts the loop, which all the files in the loop would be process with the following command. This “for” loop states what the input files will be. The “%%a” serves as a variable which will represent each file with certain file extension. In “(“*.*”)” part, users could fill with “*.mp3”, which means all the files that have the file extension of .mp3. After creating the loop, adding the “do” command in bash script followed by ffmpeg and options to complete the whole line.[29] Do not forget to change directory (cd) to the folder that contains those files.

Another major function for ffmpeg is framemd5 and framecrc, which produce one md5 or crc checksum per frame. Checksum is a hash value for detecting errors of digital files. Depend on different checksum algorithms, the checksums that are generated could be different. Those common algorithms could be CRC32, MD5, SHA-1, and SHA-256. For average text or image file, if the checksums of the files have changed, it indicates is the file is corrupted. The audio/visual files are large and have so many layers. The whole file checksum only tells if there is any change for the whole file, but it is unable to track which part of the file is wrong. In the case of audio/visual files, one checksum per file might not be effective for digital preservation practitioner. Through decoding a file and processing the decoded data

[29] “Creating Bash script to batch process with ffmpeg”. ffmprovier. AMIA Open Source. Web. Dec. 7, 2016. <http://amiaopensource.github.io/ffmprovisr/#batch_processing>

to generate a framemd5 document, each decoded audio and video frame are documented according to its timestamp, digital size, and MD5 checksum. This command will create an MD5 checksum per frame for the input file.

ffmpeg -i [input_file] -f framemd5 [output_file]

The CRC (cyclic redundancy check) checksum is more commonly adopted by ffv1 codec. ffv1 is a lossless intra-frame video codec that is a part of the codec library, libavcodec, of FFmpeg project. The video with ffv1 codec is always in Matroska container format. ffv1 codec that is in version 1.3 has mandatory CRC checksum built into each frame header. This decision helps a lossless format to be self-checking. This command of ffmpeg is for ffv1 version 1.3 to check the fixty:

ffmpeg -i input_file -f null -[30]

The "-report" could be added before the "-i" option, but it is not necessary. The "-f null-" part means video is decoded with the null muxer. The "-" at the end of the command allows video decoding without creating an output file. If there is checksum mismatch, the error will display in the terminal window. The fundamental of this command is decoding the video to perform the fixity check. Note that Final Cut Pro (FCP) does not support ffv1 or Matroska

[30] "Check FFV1 Version 3 Fixity" ffmprovisr. AMIA Open Source. Web. Dec. 7, 2016. <http://amiaopensource.github.io/ffmprovisr/#check_FFV1_fixity>

container. So, if users want to import ffv1 video into FCP, they would better transcode the file to a file with Quicktime container format.

ffmpeg: user community & resource

As a free software project, FFmpeg has large user community and many mailing lists, forums and user groups for people to discuss and exchange information. FFmpeg has many general users and they are not necessary audio/visual archivists. So, when choosing a place to consult about commands, archivists should go to the places that always solve digital preservation issues. Here are some places that are recommended:

The mailing list: https://lists.ffmpeg.org/mailman/listinfo/ffmpeg-user/. Kieren O'Leary and Dave Rice are both on this list, and they tend to answer archival questions.

FFmprovisr: http://amiaopensource.github.io/ffmprovisr/. It is a repository developed by a group of archivists during the Hack Day of Association of Moving Image Archivist (AMIA) conference in 2015. Now the website is managed by AMIA Open Source Committee.

FFmpeg Cookbook for Archivists by Reto Kromer: https://avpres.net/FFmpeg/. This is the website of audio-visual archivists Reto Kromer. He also contributes to and maintains ffmprovisr project.

ffmpeg: case study[31]

[31] This part might be restricted because I bring up things from my internship.

When I interned in Carnegie Hall archive, my supervisor Kathryn Gronsbell found the copyright metadata field for a batch of preservation master files are wrong. Instead of sending the hard drive back, Kathryn tried to change the metadata field by herself. At first, she used the simple command with the "-metadata[:metadata_specifier] key=value" option. We tested one file first, which was one of those large preservation master files. It took around an hour. When we ran Mediainfo for the output file, we found out the output file is much smaller than the original test file. Kathryn figured out the problem immediately, which was that ffmpeg's default only let it process one stream of each channel among image, audio and subtitle. Then we did the research and found out the "-map" option can disable the default. But using "-map" option was not enough in this case. Once we disabled the default, the process time took very long, which was over 24 hours even for one file. Then Kathryn found out the copy parameter under the "-codec" option, which could make ffmpeg omit the decoding and encoding step for the specified stream. And unfortunately, the copyright metadata field is the global metadata within the file, so when ffmpeg rewrote the field, it also rewrote every bits and pieces in the file. Those files that we had were Quicktime files. Since Quicktime is a proprietary file format container, it is very complicated to change the atoms. After we finally got the command correct and had one file changed, we found ffmpeg also change the other metadata fields. Through Mediainfo, it showed that the creation time for the file became "0000-00-00" and the creator for the file also came out weird. There were some bugs within ffmpeg that could not make the time correctly. This was the problem beyond our abilities to handle the tool.

FFmpeg: Issues and Future

Because FFmpeg is an open source free software project, everyone could join the community and contribute to the project. This nature of FFmpeg also incurs some issues.

There are many resources and documents in FFmpeg's official website as well as the other forums and websites. Within the project, ffmpeg draws the most attention from users. ffprobe functions as a media analyzer but it does not present some level of clarity for metadata fields. FFmpeg does have some options for ffprobe to output a comprehensive metadata report, but users need to contribute time to learn for doing that. Archivists are willing to use MediaInfo instead of ffprobe to analyze files. The same situation is also for ffplay. These two command line tools do not have as many of the users as ffmpeg. The unbalanced situation for FFmpeg is alarming. Most users do not even realize the existence of the other three command line tools within the project.

All the four tools share the same seven libraries within the project, and their documentations are similar. The documentations in FFmeg's official website are a bit messy. One of the examples is that in the general document of ffmpeg, there is a filter section, and there is also a document specifically for filters. Some of the content in these two documents repeat each other but they do not express the same thing with the same style of language. Clearly two separated people or groups contribute these two documents, and no one put effort on proof reading the documents and making them connected. Users would be confused while

reading these documents. Besides the documents on the FFmpeg website, there is also a branch called FFmpeg wiki that connected to the website. There are also some similar contents in those wiki pages but they are more advanced compared to those on the website. As a user, one could pull up a dozen web pages but has no idea which one to trust. Do not even mention that there are many grammatical mistakes for the language.

Users always go to mailing lists, forums to ask for help instead of consulting the documents on the official website. People borrow each other's command but not many of them feel it is necessary to try to build a formal repository for those commands. FFmprovisr and Reto Kromer's FFmpeg Cookbook serve as some level of repositories, but the commands are not comprehensive.

My suggestion for the future of FFmpeg is to organize a formal task force to organize rewrite the documents of FFmmpeg as well as compiling the useful commands. The maintainers of FFmpeg could organize the task force and have some experienced audio/visual archivists who are in the ffmpeg user community. Because FFmpeg is such a large project and contains so many bits and pieces, the focus could be put on ffmpeg, which has large user basis. The archivists could also contribute on expanding the FFmpeg libraries for reach what they need.

Work Cited

“About FFmpeg”. FFmpeg.. Web. Dec. 9, 2016.

< https://ffmpeg.org/about.html>.

“Check FFV1 Version 3 Fixity” ffmprovisr. AMIA Open Source. Web. Dec. 7, 2016.

<http://amiaopensource.github.io/ffmprovisr/#check_FFV1_fixity>

“Creating Bash script to batch process with ffmpeg”. ffmprovier. AMIA Open Source. Web. Dec. 7, 2016.

<http://amiaopensource.github.io/ffmprovisr/#batch_processing>

“ffmpeg Documentation 5.4 Main options” FFmpeg. Web. Dec. 9, 2016.

<https://ffmpeg.org/ffmpeg.html#Main-options>

“ffmpeg Documentation 5.11 Advanced options” FFmpeg, Web. Dec 9, 2016.

<https://ffmpeg.org/ffmpeg.html#Advanced-options>

“FFmpeg Filtering Guide” FFmpeg Wiki. Web. Dec. 9, 2016.

<https://trac.ffmpeg.org/wiki/FilteringGuide>

“ffmpeg Documentation 3.2 Stream copy” FFmpeg. Web. Dec.9, 2016.

<https://ffmpeg.org/ffmpeg.html#Stream-copy>

“FFmpeg Filters Documentation 9.166 yadif” FFmpeg. Web. Dec. 9, 2016.

<https://ffmpeg.org/ffmpeg-filters.html#yadif-1>

“ffmpeg Documentation 1 Synopsis” FFmpeg. Web. Dec. 9, 2016.

<https://ffmpeg.org/ffmpeg.html#Synopsis>

“ffmpeg Documentation 5.3 AVOptions” FFmpeg. Web. Dec. 9, 2016.

<https://ffmpeg.org/ffmpeg.html#AVOptions>

“FFmpeg Filters Documentation” FFmpeg. Web. Dec. 9, 2016.

< https://ffmpeg.org/ffmpeg-filters.html#Filtergraph-description>

“FFmpeg Filters Documentation. 9.125 Scale” FFmpeg. Web. Dec. 9, 2016.

< https://ffmpeg.org/ffmpeg-filters.html#scale-1>

“FFmpeg Scaler Documentation 2. Scaler Options” FFmpeg. Web. Dec. 9, 2016.

<https://ffmpeg.org/ffmpeg-scaler.html#scaler_005foptions>

"FFmepg" Wikipedia: The Free Encyclopedia. Wikimedia Foundation, Inc. 12 Dec. 2016.

Web. 12 Dec. 2016

< https://en.wikipedia.org/wiki/FFmpeg>

“ffmpeg Documentation”. FFmpeg. Web. Dec. 9, 2016.

< https://ffmpeg.org/about.html>

“Filter(Video)” Wikipedia: The Free Encyclopedia. Wikimedia Foundation, Inc. May 28,

2016. Web. Dec. 9, 2016.

< https://en.wikipedia.org/wiki/Filter_(video)>

“The Demultiplexer”. Electronics Tutorials. Web. Dec. 12.

< http://www.electronics-tutorials.ws/combination/comb_3.html>