

---

# Intro-to-FFMPEG-Workshop

**jycm205**

**Apr 14, 2025**



## CONTENTS:

<b>1</b>	<b>Helpful Links</b>	<b>1</b>
1.1	Resources for Archivists . . . . .	1
1.2	ffmpeg.org Links . . . . .	1
<b>2</b>	<b>Getting Started - The Command Line</b>	<b>3</b>
2.1	Accessing the Command Line . . . . .	3
2.2	What Am I Looking At? . . . . .	5
<b>3</b>	<b>Command Line Basics</b>	<b>7</b>
3.1	Navigation . . . . .	7
3.2	Wildcards . . . . .	10
3.3	Redirects and Pipes . . . . .	10
3.4	Combining Commands . . . . .	11
3.5	Loops and Batch Processing . . . . .	12
3.6	Checksumming Files . . . . .	12
<b>4</b>	<b>Installing FFmpeg</b>	<b>13</b>
4.1	Windows . . . . .	13
4.2	Mac . . . . .	18
4.3	Linux . . . . .	20
4.4	ChromeOS . . . . .	20
<b>5</b>	<b>Verifying Installation</b>	<b>21</b>
5.1	Checking the Version . . . . .	21
5.2	Bringing Up the Help Text . . . . .	22
5.3	Reading the Manual (MacOS/Linux ONLY) . . . . .	22
<b>6</b>	<b>FFmpeg Basics</b>	<b>23</b>
6.1	Basic FFmpeg Command Syntax . . . . .	23
6.2	Interpreting FFmpeg Messages . . . . .	24
<b>7</b>	<b>Generating Files</b>	<b>27</b>
7.1	Using FFplay to Preview Outputs . . . . .	27
7.2	Video . . . . .	27
7.3	Audio . . . . .	28
<b>8</b>	<b>Stream Mapping and Editing Metadata</b>	<b>29</b>
8.1	Stream Mapping . . . . .	29
8.2	Changing Color Metadata . . . . .	29
<b>9</b>	<b>Cutting and Combining Video</b>	<b>31</b>

9.1	Cut Video . . . . .	31
9.2	Combine Videos . . . . .	31
9.3	Combine Separate Audio and Video Files . . . . .	32
<b>10</b>	<b>Pad, Crop, and Scale Videos</b>	<b>33</b>
10.1	Pad . . . . .	33
10.2	Crop . . . . .	33
10.3	Scale . . . . .	33
<b>11</b>	<b>Extracting Frames and Streams</b>	<b>35</b>
11.1	Extract One Frame . . . . .	35
11.2	Extract Multiple Frames . . . . .	35
11.3	Extract All Keyframes . . . . .	35
11.4	Extract a Stream . . . . .	35
<b>12</b>	<b>Image Sequences</b>	<b>37</b>
12.1	Loop a Single Frame . . . . .	37
12.2	Image Sequence to Video . . . . .	38
<b>13</b>	<b>Using FFprobe</b>	<b>39</b>
13.1	General Syntax . . . . .	39
13.2	Specifying Fields . . . . .	39
<b>14</b>	<b>Frame and Stream MD5s</b>	<b>41</b>
14.1	FrameMD5 . . . . .	41
14.2	Stream MD5 . . . . .	41
<b>15</b>	<b>Spectrograms</b>	<b>43</b>
15.1	Using FFmpeg . . . . .	43
15.2	Using SoX . . . . .	43
<b>16</b>	<b>Indices and tables</b>	<b>45</b>

## HELPFUL LINKS

### 1.1 Resources for Archivists

[ffmprovizr](#)

[FFmpeg Cookbook for Archivists](#)

### 1.2 ffmpeg.org Links

[Manual](#)

[Main Documentation Page](#)

[FFmpeg Filters Documentation](#)

[Fancy Filtering Examples](#)

[FFmpeg Bug Tracker and Wiki](#)

[FFprobe Documentation](#)

[FFprobe Tips](#)



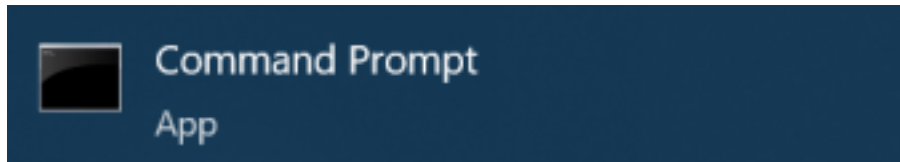
## GETTING STARTED - THE COMMAND LINE

Different operating systems will have different default applications for accessing the command line. These applications might be referred to using terms like “terminal”, “shell”, “command prompt”, or generically using the term “command line interface” (CLI). They allow you to interact with a computer using an interface where you type in strings of text in order to execute commands. This contrasts with a graphical user interface (GUI) where you are able to interact with a computer using an interface composed of buttons, menus, icons, and other visual elements.

### 2.1 Accessing the Command Line

#### 2.1.1 Windows

##### Command Prompt



Command prompt (CMD) is the basic command line interpreter for Windows computers. It has been the default on Windows for a long time. For executing basic commands, CMD will generally be enough, although it lacks some of the convenient features of other options.

- To open a Command Prompt window, search for “CMD” in the Start Menu.

##### Powershell



PowerShell is a newer program for interacting with the command line on Windows systems and is more comparable to Mac and Linux terminal programs than CMD. It has some significant structural differences and advanced features that set it apart from CMD, but is not necessarily required for anything covered here.

- To open a PowerShell window, search for “powershell” in the Start Menu.

### 2.1.2 Mac

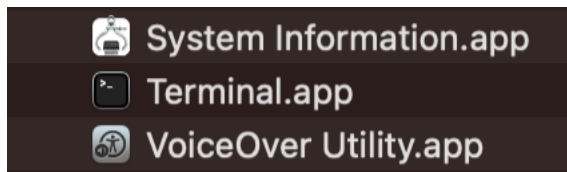
#### Terminal

The default application for interacting with the command line on Macs is called Terminal. It is similar in many ways to the default command line interpreter on many Linux distros, although some commands will still vary between the two.

- Open your Applications folder (Command + Shift + A).
- Expand the “Utilities” folder.

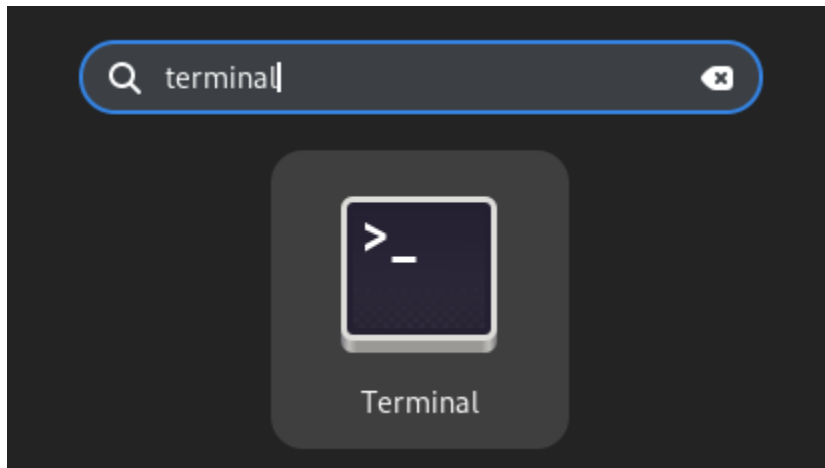


- Open the Terminal application.



### 2.1.3 Linux

#### Terminal



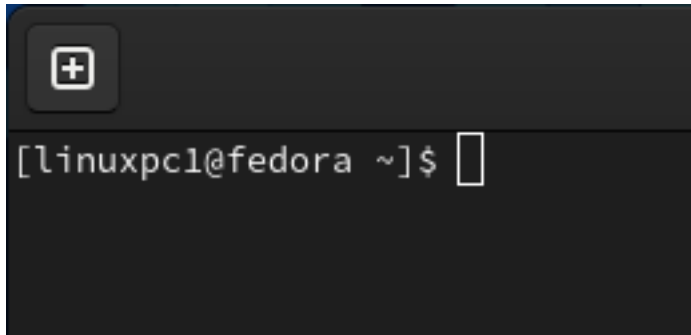
The name of your default terminal application may vary depending on the distribution you are using. Most GNOME-based distros will include GNOME Terminal as a default. Distros using a KDE desktop will most likely use Konsole as the default terminal application.



## 2.2 What Am I Looking At?

### 2.2.1 Mac/Linux

When you first open the command line, you will see something that looks similar to this on a Mac or Linux computer:

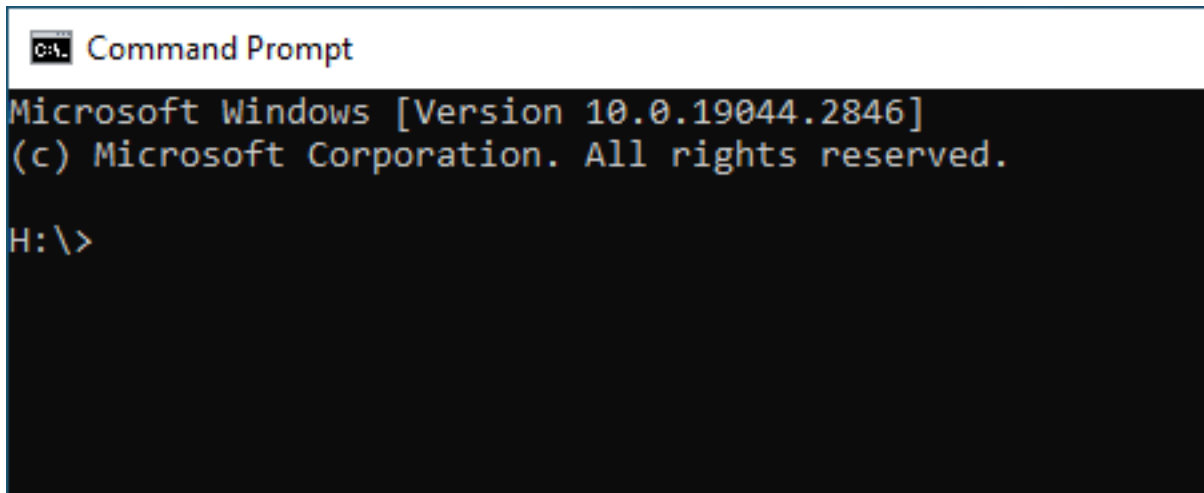


The terminal gives us a few pieces of information:

- username @ computer name
- This is followed by the folder we are currently running commands from (~ indicates our “home” directory)
- The \$ (or % on Mac computers) indicates that everything after that will be commands that we type.

### 2.2.2 Windows

On a Windows computer, the Command Prompt window will look something like this:



In this example, the part before the > tells us which folder we are currently running commands from, while the area after the > is where we'll be typing commands.



## COMMAND LINE BASICS

### 3.1 Navigation

#### 3.1.1 whoami

Prints the user that is you are currently executing commands as.

MacOS

```
whoami
```

Linux

```
whoami
```

Windows

```
whoami
```

#### 3.1.2 Print Working Directory

Prints the full path of the current “working” directory. This is where you are currently running commands from.

MacOS

```
pwd
```

Linux

```
pwd
```

Windows

```
cd
```

### 3.1.3 Change Directories

We can navigate folders on our computer from the command line using the `cd` command similar to how we would in a GUI file browser.

MacOS

Change directories to Downloads folder.

```
cd /home/myUsername/Downloads
```

Change directories to current User's home directory.

```
cd ~
```

Go back one folder level.

```
cd ..
```

Linux

Change directories to Downloads folder.

```
cd /home/myUsername/Downloads
```

Change directories to current User's home directory.

```
cd ~
```

Go back one folder level.

```
cd ..
```

Windows

Change directories to Downloads folder.

```
cd C:\Downloads
```

When changing between different drives, include `/d`.

```
cd /d D:\Directory
```

Go back one folder level.

```
cd ..
```

### 3.1.4 List Folder Contents

Lists the contents of the current working directory. If you follow the command with the path to a different directory, it will list the contents of that directory instead.

MacOS

List current folder contents

```
ls
```

List contents of Documents folder

```
ls "/home/myUsername/Documents"
```

Linux

List current folder contents

```
ls
```

List contents of Documents folder

```
ls "/home/myUsername/Documents"
```

Windows

List current folder contents

```
dir
```

List contents of My Documents folder

```
dir "C:\Users\myUsername\Documents"
```

### 3.1.5 Notes About Quotes

The command line uses spaces to separate various parts of commands. As a result, file and folder paths that contain spaces need to be handled in special ways to ensure that those spaces are correctly read as part of a larger string of text rather than separators within the command. A common way to accomplish this is to enclose the path in single ( ' ') or double ( " ") quotes.

MacOS

```
cd "/home/myUsername/path with spaces"
```

Linux

```
cd "/home/myUsername/path with spaces"
```

Windows

```
cd "C:\Users\myUsername\path with spaces"
```

- Another way to handle spaces is by “escaping” them using a special character ( \ ) that tells the command line to ignore its usual interpretation of whatever immediately follows that character. This might look something like `/home/myUsername/path\ with\ spaces`
- You’ll notice that when you drag and drop files with spaces in the path or filename onto the command line, the resulting path will always be formatted in a way that handles the spaces appropriately (either enclosing the text in quotes or inserting escape characters before each space).
- Be careful when copying quoted paths/filenames from certain sources, such as Word documents or spreadsheets, as these types of documents may format text in ways that can cause issues for commands. For example, the following single ( ' ') and double ( " ") quotes will NOT work.

### 3.2 Wildcards

Certain special characters (?, !, [], \*) are used to match patterns. This can be very useful when trying to run a command on a certain subset of files or folders that share certain identifying characteristics. One example of a very common wildcard is \*, which matches any combination of characters.

MacOS

List all .mp4 files in the current directory

```
ls *.mov
```

List all .mp3 files in every folder in the Music directory

```
ls /home/myUsername/Music/*/*.mp3
```

Linux

List all .mp4 files in the current directory.

```
ls *.mp4
```

List all .mp3 files in every folder in the Music directory.

```
ls /home/myUsername/Music/*/*.mp3
```

Windows

List all .mp4 files in the current directory.

```
dir *.mp4
```

List all .mp3 files in every folder in the Music directory.

```
dir C:\Users\myUsername\Music\*/*.mp3
```

### 3.3 Redirects and Pipes

#### 3.3.1 Pipe

Pipes (|) are used to pass the output from one command to another program.

#### 3.3.2 Redirect

Redirects (>, < or >>, << to append) are used to pass the output from one command to a file or stream.

MacOS

```
ls > file_list.txt
```

Linux

```
ls > file_list.txt
```

Windows

```
dir > file_list.txt
```

## 3.4 Combining Commands

### 3.4.1 Performing Commands Sequentially

Commands can be set to execute one after another. This allows us to queue up a number of commands to run in sequence.

MacOS

```
command1 ; command2
```

Linux

```
command1 ; command2
```

Windows

```
command1 & command2
```

### 3.4.2 Conditionally Run Commands

Commands can be set to run depending on whether the previous command succeeded (exit status = 0) or failed (non-zero exit status). This can allow us to make sequential commands that are more interdependent. If a later command depends on the output from a previous command being created successfully, we can ensure that our second command only runs if that first command successfully created the necessary output. Similarly, can build in error handling to divert commands down a different path if a previous command failed.

MacOS

Run command2 only if command1 completed successfully

```
command1 && command 2
```

Run command2 only if command1 failed

```
command1 || command2
```

Linux

Run command2 only if command1 completed successfully

```
command1 && command2
```

Run command2 only if command1 failed

```
command1 || command2
```

Windows

Run command2 only if command1 completed successfully

```
command1 && command2
```

Run command2 only if command1 failed

```
command1 || command2
```

## 3.5 Loops and Batch Processing

### 3.5.1 For Loops

MacOS

```
for x in /home/myUsername/example/*.flac ; do ffmpeg -i "$x" -ab 320k -f mp3 "${x//.flac}~  
↪".mp3 ; done
```

Linux

```
for x in /home/myUsername/example/*.flac ; do ffmpeg -i "$x" -ab 320k -f mp3 "${x//.flac}~  
↪".mp3 ; done
```

Windows

```
for "C:\Users\myUsername\example\" %x in (*.flac) do ffmpeg -i "%x" -ab 320k -f mp3 "%~  
↪nx.mp3"
```

## 3.6 Checksumming Files

Many operating systems include some default program that can create checksums directly from the command line.

MacOS

```
md5 input_file
```

Linux

```
md5sum input_file
```

Windows

```
certUtil -hashfile input_file MD5
```



## INSTALLING FFMPEG

For the most up-to-date instructions on how to install FFmpeg on your operating system, you should check the [FFmpeg Official Website](#).

### 4.1 Windows

#### 4.1.1 Downloading

- The current builds for Windows are provided by [gyan.dev](#).
- Follow the link from [FFmpeg's Downloads page](#).
- It is generally recommended to use the latest “git master build” of FFmpeg.
- The “full” build will have more libraries enabled than “essentials”, so download the “full” build.

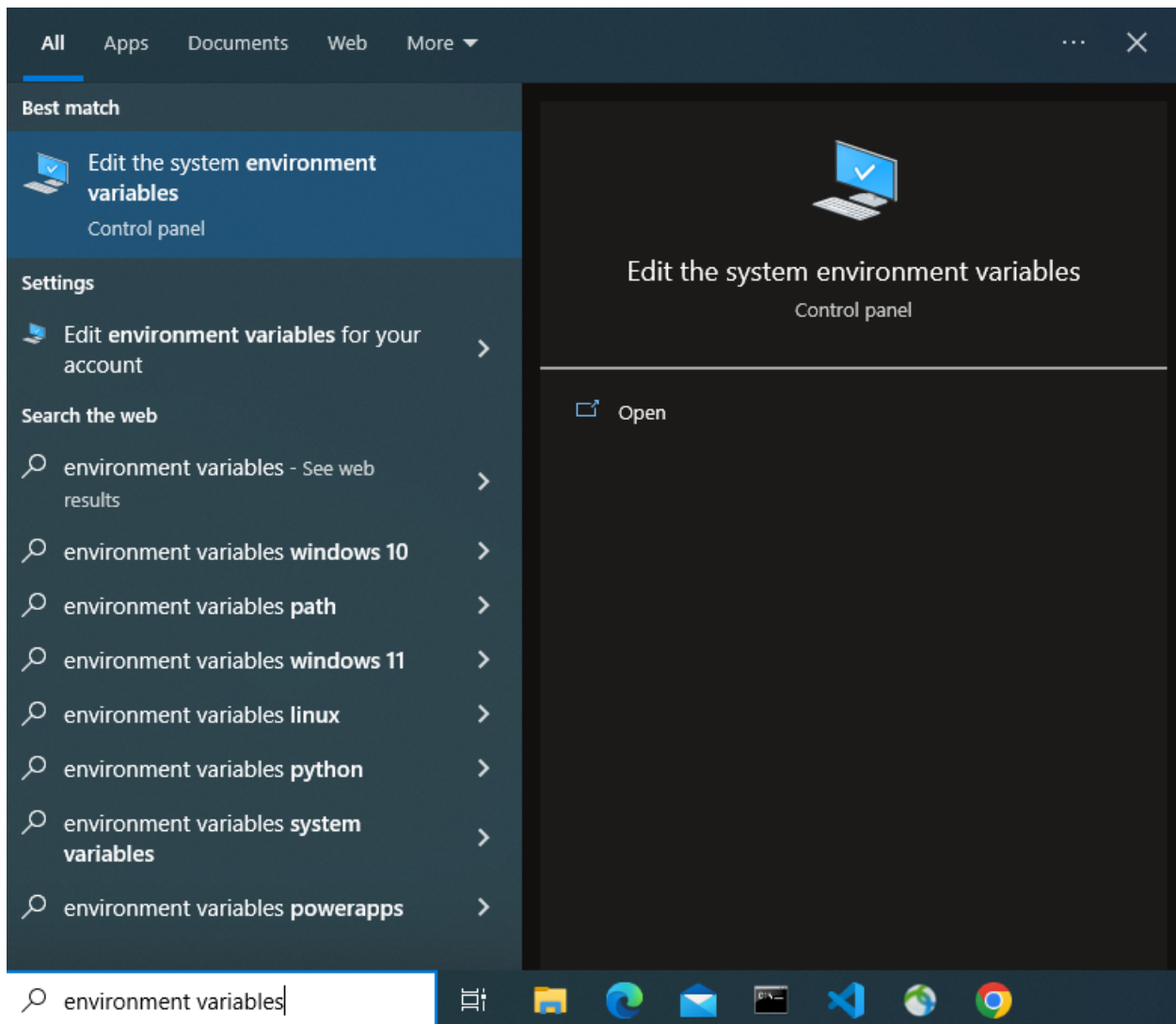


- Once the download is complete, extract the files (you will need to have 7-Zip installed to extract the .7z files. Alternatively, .zip files are also available [on github](#)).
- Inside the extracted folder, you will see a folder named “bin” that contains the executables for FFmpeg.
- If you just double click these files, a command prompt window will briefly open, then close.
- For now, if you want to run FFmpeg, you’ll either need to drag and drop the “ffmpeg” file in the “bin” folder onto the command line or use the command line to navigate to the “bin” folder and run FFmpeg commands from there.

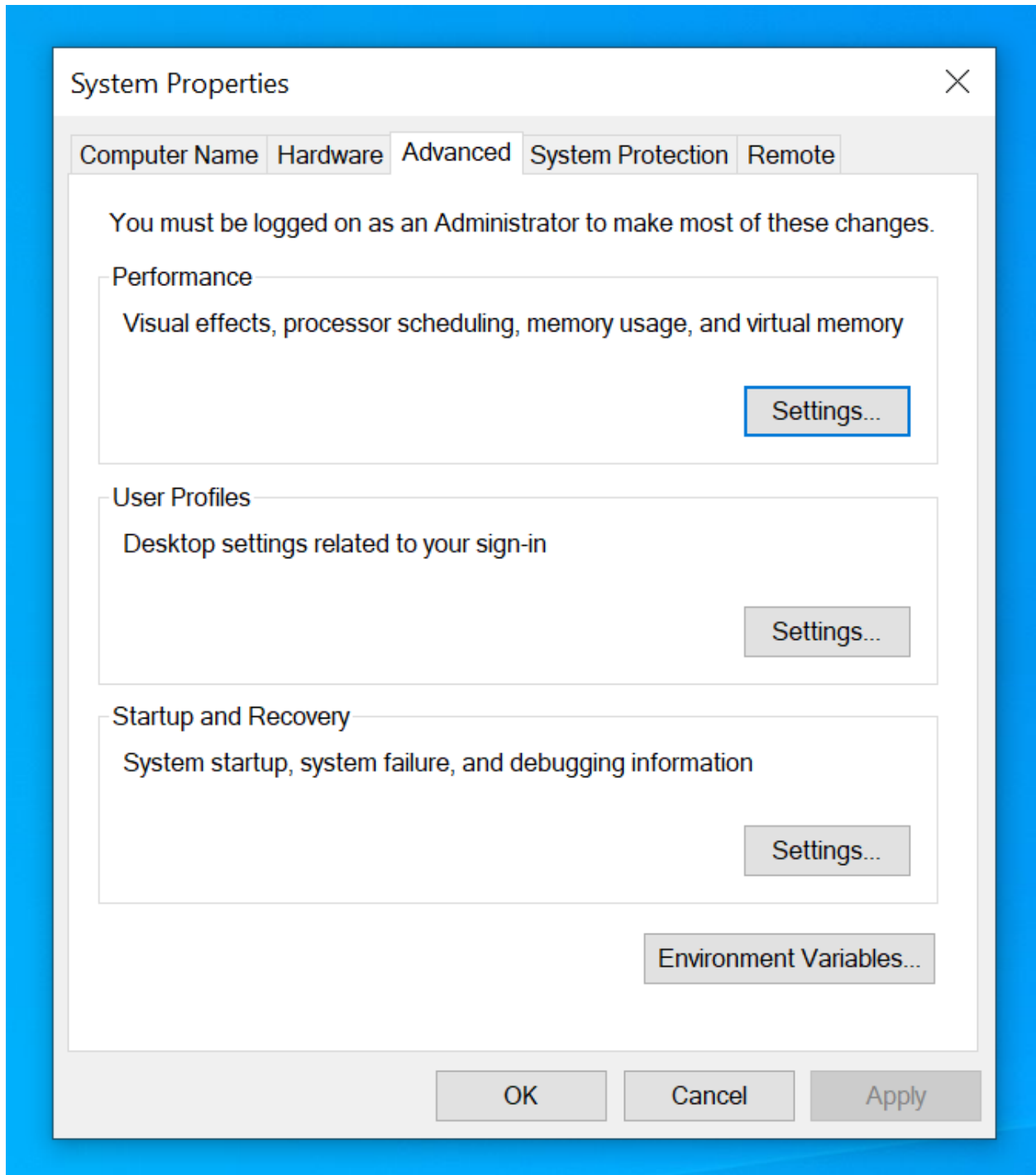
### 4.1.2 Adding FFmpeg to Path

In order for FFmpeg to be easily callable by simply typing `ffmpeg` into the command line, we'll need to add it to the "Path" for your username.

- Start by placing the contents of the FFmpeg bin folder in an easy to remember location, such as a dedicated "Applications" folder in your "C" drive.
- Next search for "environment variables" in the search bar and select the option "Edit the system environment variables".



- This will open the System Properties window.
- In this window, click the "Environment Variables" button near the bottom.

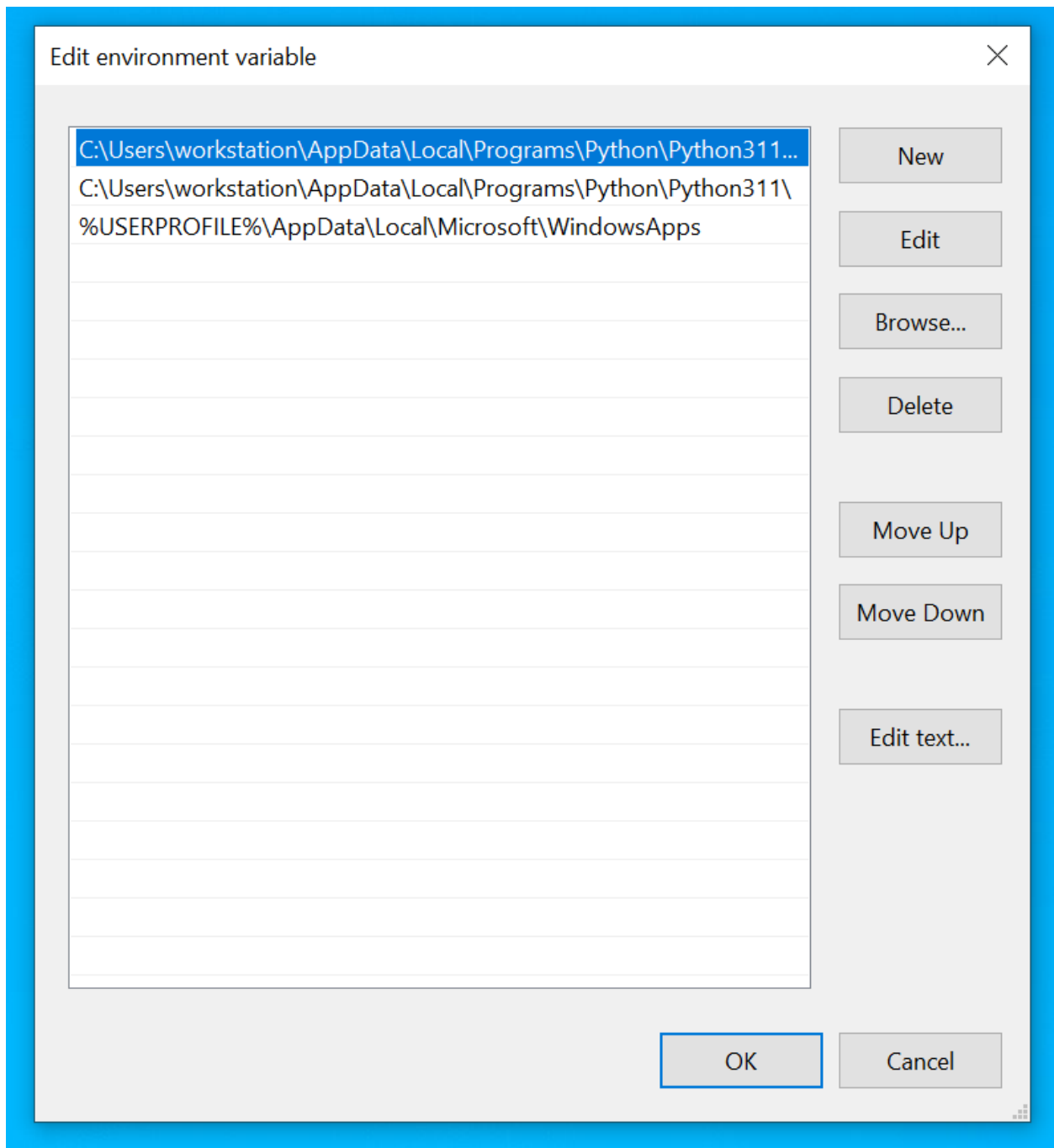


- This will open the “Environment Variables” window.
- In the top section of this window, you’ll notice an entry in the “Variable” column called “Path”.
- Double click the “Path” entry, or select it and then click the “Edit” button below.

OneDrive	C:\Users\workstation\OneDrive
Path	C:\Users\workstation\AppData\Local\Programs\Python\Python3...
TEMP	C:\Users\workstation\AppData\Local\Temp
TMP	C:\Users\workstation\AppData\Local\Temp

New...Edit...Delete

- This will open the “Edit environment variable” window.
- In this new window, click the “New” button.
- Now enter the path to the folder that contains your FFmpeg folder (for example, if you put your files in an “Applications” folder in your “C” drive, you would enter “C:\Applications”).



- To verify that the installation works, continue on to *section 5*.

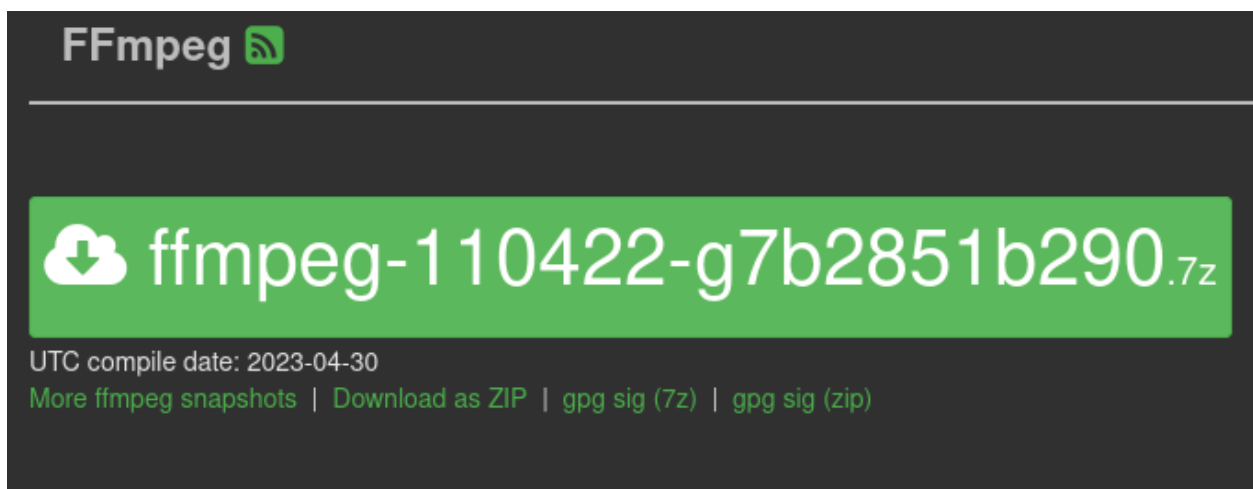
## 4.2 Mac

There are several options for installing FFmpeg on Mac computers







### 4.2.1 Option 1 - Install a Static Build

#### Downloading

- Follow the Static Build Download link from the [FFmpeg website](#).
- This will take you to a new page where you can download FFmpeg, FFprobe, and FFplay separately.
- FFmpeg recommends downloading the latest snapshots over the releases, so click the green button on the left with the a long name that looks something like: ffmpeg-(some long string of numbers and letters).7z.



- Scroll down and repeat this for the FFplay and FFprobe download links on the page as well.
- Once all the files have finished downloading, navigate to your Downloads folder and extract each of the “.7z” files by double clicking them.

Name	
	ffprobe
	ffplay
	ffmpeg
	ffplay-110011-gf456c192d9.7z
	ffprobe-110422-g7b2851b290.7z
	ffmpeg-110422-g7b2851b290.7z

- You should now have an “ffmpeg”, “ffprobe”, and “ffplay” file in your downloads folder. You can drag and drop these files into a Terminal window as is to run them, but if we want them to be easily accessible on the command line they’ll need to be added to your \$PATH.

## Adding FFmpeg to Path

- Start by opening your “Applications” folder, navigating to the “Utilities” folder, and opening the Terminal application.
- In the Terminal, run the following command (copy and paste the command, then press Enter/Return) to change directories to your Downloads folder where the “ffmpeg”, “ffprobe”, and “ffplay” files are located.

```
cd ~/Downloads/
```

- At this point, you can verify that you’re in the right location by running the `ls` command as described in Section 3.1.4. When trying to run the command you may get a prompt asking if you want to allow Terminal to access your downloads folder. Select “Yes” and the command will run. You should see the FFmpeg files that you downloaded and extracted listed in the output if you are in the right location.
- Now that you are sure you are in the right directory, it’s time to make the folder “/usr/local/bin” so that we’ll be able to move our files there. Run the following command:

```
sudo mkdir -p /usr/local/bin/
```

- You will be prompted for your password and should not see any additional messages if the command ran successfully.
- Now that the folder has been created, we simply need to copy our files there. Run the following command:

```
sudo cp ffmpeg ffprobe ffplay /usr/local/bin
```

- If you want to make sure that you successfully copied the files, you can run this command to list the contents of “/usr/local/bin”:

```
ls /usr/local/bin
```

- You should see your FFmpeg files listed in the output.
- To verify that the installation works, continue on to [section 5](#).

## Uninstalling

- To uninstall the FFmpeg files, you simply need to delete the files from the folder we put them in.
- Start by opening a Terminal window and changing directories to “/usr/local/bin”.

```
cd /usr/local/bin
```

- Check that you are in the right location by running the `ls` command. You should see the “ffmpeg”, “ffprobe”, and “ffplay” files listed in the output.
- Once you have confirmed that you are in the right location, run the `rm` command as a superuser to delete the “ffmpeg” file (NOTE: Be careful whenever running the `rm` command as a superuser. Make sure that you are only deleting the files you want to delete before running the command). The command will prompt you for your password before running.

```
sudo rm ffmpeg
```

- If the command completes successfully, you can now do the same for the “ffprobe” and “ffplay” files.

```
sudo rm ffprobe  
sudo rm ffplay
```

### 4.2.2 Option 2 - Use a Package Manager

- For an experience similar to Linux you can use a third party package manager for Macs like [Homebrew](#) or [MacPorts](#).
- Further information on this process can be found [HERE](#).

## 4.3 Linux

- FFmpeg can be installed using your operating system's package manager in most cases.
- On some Linux distributions, you may need to enable non-free repositories before you can install FFmpeg.
- For running the most recent or specific versions of FFmpeg, static builds are also an option.
- To verify that the installation works, continue on to [section 5](#).

## 4.4 ChromeOS

- NOTE - NEED TO TEST
- ChromeOS is based on Linux and should, in theory, be able to install FFmpeg using a .deb file
- Static builds for Linux may also work on ChromeOS(?)
- To verify that the installation works, continue on to [section 5](#).



## VERIFYING INSTALLATION

To verify that FFmpeg is working as expected, we'll run a couple of simple commands to check that we can successfully call the program and look at what those commands tell us.

### 5.1 Checking the Version

- Let's start by checking the version of FFmpeg we have installed.
- Try running the following command:

```
ffmpeg -version
```

- If the command worked, you should see something like this:

```
ffmpeg version 5.1.3 Copyright (c) 2000-2022 the FFmpeg developers
built with gcc 12 (GCC)
configuration: --prefix=/usr --bindir=/usr/bin --datadir=/usr/share/ffmpeg --docdir=/usr/share/doc/ffmpeg --incdir=/usr/include/ffm
peg --libdir=/usr/lib64 --mandir=/usr/share/man --arch=x86_64 --optflags='-O2 -flto=auto -ffat-lto-objects -fexceptions -g -grecord
-gcc-switches -pipe -Wall -Werror=format-security -Wp,-D_FORTIFY_SOURCE=2 -Wp,-D_GLIBCXX_ASSERTIONS -specs=/usr/lib/rpm/redhat/redh
at-hardened-cc1 -fstack-protector-strong -specs=/usr/lib/rpm/redhat/redhat-annobin-cc1 -m64 -mtune=generic -fasynchronous-unwind-ta
bles -fstack-clash-protection -fcf-protection' --extra-ldflags='-Wl,-z,relro -Wl,--as-needed -Wl,-z,now -specs=/usr/lib/rpm/redhat/
redhat-hardened-ld -specs=/usr/lib/rpm/redhat/redhat-annobin-cc1 -Wl,--build-id=sha1 ' --extra-cflags='-I/usr/include/av1' --ena
ble-libopencore-amrnb --enable-libopencore-amrwb --enable-libvo-amrwbenc --enable-version3 --enable-bzlib --enable-chromaprint --di
sable-crystalhd --enable-fontconfig --enable-frei0r --enable-gcrypt --enable-gnutls --enable-ladspa --enable-libaom --enable-libdav
id --enable-libbass --enable-libbluray --enable-libbs2b --enable-libcdio --enable-libdrm --enable-libjack --enable-libjxl --enable-l
ibfreetype --enable-libfribidi --enable-libgsm --enable-libilbc --enable-libmp3lame --enable-libmysofa --enable-nvenc --enable-open
al --enable-openc1 --enable-opengl --enable-libopenjpeg --enable-libopenmpt --enable-libopus --enable-libpulse --enable-libplacebo
--enable-librsvg --enable-librav1e --enable-librubberband --enable-libsmbclient --enable-version3 --enable-libsndio --enable-libso
xr --enable-libspeex --enable-libssh --enable-libsvtav1 --enable-libtesseract --enable-libtheora --enable-libtwolam
e --enable-libvorbis --enable-libv4l2 --enable-libvidstab --enable-libvmaf --enable-version3 --enable-vapoursynth --enable-libvp8 --
enable-vulkan --enable-libshaderc --enable-libwebp --enable-libx264 --enable-libx265 --enable-libxvid --enable-libxml2 --enable-li
bzing --enable-libzmq --enable-libzvbi --enable-lv2 --enable-avfilter --enable-libmodplug --enable-postproc --enable-pthreads --dis
able-static --enable-shared --enable-gpl --disable-debug --disable-stripping --shlibdir=/usr/lib64 --enable-lto --enable-libmfx --e
nable-runtime-cpudetect
libavutil      57. 28.100 / 57. 28.100
libavcodec     59. 37.100 / 59. 37.100
libavformat    59. 27.100 / 59. 27.100
libavdevice    59.  7.100 / 59.  7.100
libavfilter     8. 44.100 /  8. 44.100
libswscale     6.  7.100 /  6.  7.100
libswresample  4.  7.100 /  4.  7.100
libpostproc   56.  6.100 / 56.  6.100
```

- Looking at the first line of the output, we can see that I have FFmpeg version 5.1.3 installed.
- The large block of text following this tells us how our particular install of FFmpeg is configured.
- Some functions require that FFmpeg be configured with certain options.
- For example, if we look at the output, we can see that `--enable-libsoxr` is listed in my configuration section. This means that FFmpeg can use the SoX Resampler library for audio.

## 5.2 Bringing Up the Help Text

- Next let's bring up the help text for FFmpeg:

```
ffmpeg -hide_banner -help
```

- You should see a list of commands, each with a short description of what it does.
- In addition to the `-help` command, we also used the `-hide_banner` command here. This is one of several commands that impacts the amount of information that FFmpeg outputs to the command line. In this case, the command simply keeps FFmpeg from printing the version and configuration information at the top of its output.

## 5.3 Reading the Manual (MacOS/Linux ONLY)

- Finally, lets bring up the manual for FFmpeg:

```
man ffmpeg
```

- You can press `q` to exit the manual.
- PowerShell and Command Prompt do not have an exact equivalent of the `man` command. However, you can still access the [full FFmpeg manual online](#).

## FFMPEG BASICS

### 6.1 Basic FFmpeg Command Syntax

FFmpeg command structure:

```
ffmpeg [global_options] {[input_file_options] -i input} {[output_file_options] output}
```

A simple FFmpeg command for transcoding to an H.264 file with AAC audio might look something like this:

```
ffmpeg -i input_file -c:v libx264 -crf 18 -preset slow -c:a aac -b:a 256k output_file
```

Let's look at what each part of this command does.

**ffmpeg**

- This is where we tell the command line which program we want to run.
- In this case we are calling FFmpeg.
- If you have one of the static builds of FFmpeg, you could also use the full path to that static build file here (just drag and drop the “ffmpeg” file on the command line!) or cd to that directory and run commands from there.

**-i input\_file**

- Input files are indicated using **-i**.
- **input\_file** is just a generic stand-in to tell us that, whatever your input file is, it should go here. If you wanted to run this command on an actual file, you would need to replace this with the path to the particular file that you want to transcode (just drag and drop it on the command line).
- There are some cases where you may want to specify multiple inputs in a single command. You can repeat the **-i** command multiple times to accomplish this (i.e. **-i input\_1.mov -i input\_2.wav**).

**-c:v libx264 -crf 18 -preset slow**

- **-c:v** indicates that we are specifying video codec settings that we want to use for our output.
- This is followed by the video encoder that we want to use. In this example we are using **libx264**, which is the name that FFmpeg uses to call the free and open source H.264 encoder library, x264.
- Some codecs may have multiple encoders available to choose from. For example, FFmpeg has three encoders to choose from for ProRes (**prores**, **prores\_ks** and **prores\_aw**).
- **-crf 18 -preset slow** are encoding settings specific to the x264 encoding library that configure some of the compression settings for the output.
- A detailed explanation of these settings as well as other settings that we could have used here can be found on the [FFmpeg Encoding wiki's H.264 page](#).

- Encoding settings will vary from one codec to another, so it's always a good idea to read the documentation and look at some examples to get a better idea of the exact settings you'll want to use.

`-c:a aac -b:a 256k`

- `-c:a` indicates that we are specifying audio codec settings that we want to use for our output. Notice that this is similar to the structure of what we had done for the video codec, but with a different codec library and different settings.
- In this case we are using the aac encoder for the audio.
- `-b:a 256k` are encoding settings for the AAC encoding library. These settings tell FFmpeg to use a constant bitrate of 256kB/s for the audio.
- A detailed explanation of these settings as well as other settings that we could have used here can be found on the [FFmpeg Encoding wiki's AAC page](#). Note that this example uses the native FFmpeg AAC encoder, but FFmpeg can also be configured with another AAC encoder, `libfdk_aac`.
- If our input video had no audio streams we could leave `-c:a aac -b:a 250k` out of the command entirely since there would be no audio to encode.
- If we wanted to copy the audio from our `input_file` without re-encoding, we could have used `-c:a copy` instead.
- If we had left out any audio codec settings, FFmpeg would have chosen a default setting for us based on the output format we specified.

`output_file`

- This part of the command specifies the full path to our output file.
- Similar to how `input_file` had been used as a generic stand-in for the input, this is just a generic stand-in for an output file that needs to be replaced with a real output file path if we actually want to run this command. In this case, we might output an H.264 file with AAC audio to a `.mp4` or `.mov` file (i.e. `/Users/myUserName/Videos/example.mp4`).
- Unlike the input, our output file does not need to be indicated with any kind of flag like `-i`. FFmpeg automatically assumes that this is our output based on the fact that it appears at the end of the command.

## 6.2 Interpreting FFmpeg Messages

By default FFmpeg will output a lot of information to the command line as it processes files.

## 6.2.1 Progress

```

frame=23103 fps=145 q=-0.0 size=      0kB time=00:16:02.62 bitrate=  0.0kbits/
frame=23175 fps=145 q=-0.0 size=      0kB time=00:16:05.62 bitrate=  0.0kbits/
frame=23247 fps=145 q=-0.0 size=      0kB time=00:16:08.62 bitrate=  0.0kbits/
frame=23319 fps=145 q=-0.0 size=      0kB time=00:16:11.62 bitrate=  0.0kbits/
frame=23393 fps=145 q=-0.0 size=      0kB time=00:16:15.01 bitrate=  0.0kbits/
frame=23468 fps=145 q=-0.0 size=      0kB time=00:16:18.00 bitrate=  0.0kbits/
frame=23542 fps=145 q=-0.0 size=      0kB time=00:16:21.01 bitrate=  0.0kbits/
frame=23616 fps=145 q=-0.0 size=      0kB time=00:16:24.00 bitrate=  0.0kbits/
frame=23687 fps=145 q=-0.0 size=      0kB time=00:16:27.00 bitrate=  0.0kbits/
frame=23761 fps=145 q=-0.0 size=      0kB time=00:16:30.04 bitrate=  0.0kbits/
frame=23842 fps=145 q=-0.0 size=      0kB time=00:16:33.41 bitrate=  0.0kbits/
frame=23919 fps=145 q=-0.0 size=      0kB time=00:16:36.62 bitrate=  0.0kbits/
frame=23985 fps=145 q=-0.0 size=      0kB time=00:16:39.37 bitrate=  0.0kbits/
frame=24062 fps=145 q=-0.0 size=      0kB time=00:16:42.58 bitrate=  0.0kbits/
frame=24129 fps=145 q=-0.0 size=      0kB time=00:16:45.37 bitrate=  0.0kbits/
frame=24207 fps=145 q=-0.0 size=      0kB time=00:16:48.62 bitrate=  0.0kbits/
frame=24279 fps=145 q=-0.0 size=      0kB time=00:16:51.62 bitrate=  0.0kbits/
frame=24351 fps=145 q=-0.0 size=      0kB time=00:16:54.62 bitrate=  0.0kbits/
frame=24423 fps=145 q=-0.0 size=      0kB time=00:16:57.62 bitrate=  0.0kbits/
frame=24495 fps=145 q=-0.0 size=      0kB time=00:17:00.62 bitrate=  0.0kbits/
frame=24567 fps=145 q=-0.0 size=      0kB time=00:17:03.62 bitrate=  0.0kbits/
frame=24639 fps=145 q=-0.0 size=      0kB time=00:17:06.62 bitrate=  0.0kbits/
frame=24711 fps=145 q=-0.0 size=      0kB time=00:17:09.62 bitrate=  0.0kbits/
s speed=6.04x

```

- FFmpeg will start by outputting the banner with information about the FFmpeg version and how it is configured.
- This will be followed by information about the input and output files you have set with your command.
- As FFmpeg processes the encoding, it will output a regularly updating string with information about the fps that the encoding process is running at, bitrate, etc.
- This output can be useful for judging how long processing might take or how large your output file might be.
- If you want to stop an encoding after it has started you can press “q” to interrupt the process.
- FFmpeg may output other information such as errors or warnings to the command line as well.

## 6.2.2 Warnings

- Warnings are indicated by yellow text in FFmpeg’s output.
- Generally warnings provide you with information about things that *may* cause issues, but do not prevent an operation from running.
- Many warnings are benign, but it is always a good idea to look up what a warning means if you aren’t sure.
- For example, unsupported timecode streams in an input will often cause warnings.

### 6.2.3 Errors

- Errors are indicated by red text in FFmpeg's output.
- You will typically encounter errors when certain settings are used incorrectly. Often these will prevent an operation from running.
- For example, FFmpeg will output an error message if you try to use `-c:v copy` with a video filter (`-vf`) because these two settings contradict one another.

### 6.2.4 Changing Verbosity

- FFmpeg, FFprobe, and FFplay have various settings for adjusting the amount of information that gets output to the command line as they runs.
- For example, in the Verifying Installation section we used `-hide_banner` to suppress the banner that FFmpeg typically outputs when it runs.
- The `-loglevel` command can also be used to adjust the verbosity of the output (`-v` can also be used in place of `-loglevel`). You can run the command `ffmpeg -loglevel -h` to see a list of options. Documentation about these settings can also be found [HERE](#).
- `ffmpeg -loglevel quiet -i input_file output_file` can be used to prevent FFmpeg from writing anything to the command line while running.

### 6.2.5 Overwriting Files

- If your output file already exists, FFmpeg will prompt you asking if you want to overwrite the existing file before it proceeds.
- If you type `y` and press enter, FFmpeg will overwrite the existing file. If you type `n` and press enter, FFmpeg will stop the command you are trying to run.
- You can tell FFmpeg to automatically select “yes” using the `-y` flag before your input (i.e. `ffmpeg -y -i input_file output_file`).

## GENERATING FILES

FFmpeg can generate many different patterns using its filtergraph functionality. Generating signals and patterns uses the `lavfi` command.

### 7.1 Using FFplay to Preview Outputs

FFplay can be a useful tool for checking what the output will look/sound like before writing it to a file.

```
ffplay -f lavfi -i {[filter] [parameters]}
```

### 7.2 Video

#### 7.2.1 Mandelbrot Pattern

```
ffmpeg -f lavfi -i mandelbrot=size=1280x720:r=24 -t 60 -c:v libx264 -crf 26 -preset fast -profile:v main -pix_fmt yuv420p output_file.mov
```

Documentation about adjusting parameters for the Mandelbrot pattern can be found in [section 15.6 of the FFmpeg filters documentation](#).

ffmpeg

- Call FFmpeg

`-f lavfi -i mandelbrot=size=1280x720:r=24`

- Here we are passing the `mandelbrot` filter as our input.
- We are including a few other settings (`size` and `r`) to set the size and framerate as well.

`-t 60`

- This sets the time/duration of our video in seconds.

`-c:v libx264`

- This is setting `libx264` as the encoder that we'll use.

`-crf 26 -preset fast -profile:v main`

- This is configuring the compression settings for `libx264`.

`-pix_fmt yuv420p`

- This sets the pixel format of our output video. This includes things like the chroma subsampling (4:2:0 in this case).

output\_file.mp4

- This needs to be the full path to the output file that we want to create.

### 7.2.2 Test Patterns

The same command structure can be used and modified to fit your needs.

Here is a short list of some test patterns that FFmpeg can output: testsrc, testsrc2, smptebars, smptehdbars, pal75bars, colorspectrum

Documentation on the available test patterns that FFmpeg can generate can be found in [section 15.10 of the FFmpeg filters documentation](#).

## 7.3 Audio

### 7.3.1 Noise Generator

FFmpeg can generate various types of noise using the anois-src filter. Documentation on this filter can be found in [section 9.8 of the FFmpeg filters documentation](#).

```
ffmpeg -f lavfi -i anois-src=color=pink:duration=100:sample_rate=48000 -af volume=-18dB -  
-c:a pcm_s16le output_file.wav
```

The structure of this command is similar to the previous video examples, but uses some additional audio-specific settings, such as sample\_rate to set the audio sample rate and -af to set the volume. In this case, we are also outputting to an uncompressed 16-bit PCM WAV file.

### 7.3.2 Sine Wave Generator

FFmpeg's sine wave generator is another filter that can be used to produce sine wave patterns at different frequencies. Documentation on this filter can be found in [section 9.11 of the FFmpeg filters documentation](#).

```
ffmpeg -f lavfi -i sine=frequency=1000:duration=100:sample_rate=48000 -c:a pcm_s16le -  
-o output_file.wav
```



## STREAM MAPPING AND EDITING METADATA

Streams can be copied when performing actions that do not require re-encoding. This includes things like changing metadata, removing or adding streams, or cutting/combining videos on intra-compressed frames.

Rather than specifying an encoder for the output, we can simply use `-c:v copy` to copy a video stream and `-c:a copy` to copy an audio stream. `-map 0 -c copy` will also copy all streams.

### 8.1 Stream Mapping

The `-map` command is useful for singling out particular streams within complex video files. Documentation about the `-map` command can be found [HERE](#).

`-map 0`

This is used to indicate all streams from the first input.

`-map 0:a:1`

- The first number in the map command indicates the input, with `0` corresponding to the first input. If passing multiple inputs to FFmpeg in a single command, the `-map` command can be repeated to map streams from various inputs into the final output.
- The following letter identifies the type of stream (a for audio, v for video, s for subtitle, etc.)
- The final number indicates that stream's number for that type of stream within the container, starting from `0`. For example, `a:1` would be the second audio stream in the file.

### 8.2 Changing Color Metadata

The sample video files created in Section 7 did not include any commands to tag how the color should be decoded in the videos. Adding this information now does not require re-encoding the files.

The following command will update the Color Range, Color Primaries, Transfer Characteristics, and Color Space to be read as limited range Rec. 709 video in addition to re-wrapping the H.264 encoding in an mp4 wrapper.

```
ffmpeg -i input_file.mov -c:v copy -color_range tv -colorspace bt709 -color_trc bt709 -  
↪color_primaries bt709 output_file.mp4
```



## CUTTING AND COMBINING VIDEO

### 9.1 Cut Video

Cutting video can be done several different ways using FFmpeg. Note that you can only copy video streams if cutting on intra-compressed frames.

#### 9.1.1 Define a Duration

You can define a start point for the video and then a duration for how long the video should run using the following command.

```
ffmpeg -i input.mkv -ss 00:01:00 -t 00:00:05 -c copy output.mp4
```

#### 9.1.2 Define a Timecode Range

You can define a range of timecodes using the following command.

```
ffmpeg -i input.mkv -ss 00:01:00 -to 00:01:05 -c copy output.mp4
```

#### 9.1.3 Cutting from the End of the File

You can use the following command to seek backwards from the end of the file to the point that you want to cut as well. Note that this takes a negative time value.

```
ffmpeg -i input.mkv -sseof -00:01:00 -c copy output.mp4
```

### 9.2 Combine Videos

See [documentation](#).

There are several ways to combine videos with FFmpeg depending on your inputs:

- concat demuxer: Used to losslessly combine codecs such as H.264. Files should have the same aspect ratio.
- concat protocol: Formats such as DV, MPEG-1, and MPEG-2 can be combined with the concat protocol.
- concat video filter: Used when combining different codecs.

## 9.2.1 Using the Concat Demuxer

The concat demuxer takes a list of files as an input and combines them to create the output. The text list uses the following structure:

```
file 'file1.mp4'
file 'file2.mp4'
```

The file list can be manually created, or automatically created using a command similar to this:

MacOS

```
for f in *.mp4 ; do echo "file '$f'" >> mylist.txt ; done
```

Linux

```
for f in *.mp4 ; do echo "file '$f'" >> mylist.txt ; done
```

Windows

If using CMD:

```
(for %i in (*.mp4) do @echo file '%i') > mylist.txt
```

If using Powershell:

```
foreach ($i in Get-ChildItem .\*.mp4) {echo "file '$i'" >> mylist.txt}
```

Once the text list is created, the files can be combined with the following FFmpeg command:

```
ffmpeg -f concat -safe 0 -i mylist.txt -c copy output.mp4
```

## 9.3 Combine Separate Audio and Video Files

-map can be used to combine streams from different sources into a single output file.

```
ffmpeg -i input_0.mp4 -i input_1.mp4 -c copy -map 0:v:0 -map 1:a:0 out.mp4
```

If the two files have different durations, the -shortest command can be used to tell FFmpeg to stop when the shorter of the two streams ends.

```
ffmpeg -i input_0.mp4 -i input_1.mp4 -c copy -map 0:v:0 -map 1:a:0 -shortest out.mp4
```

Note that the output container must be compatible with all of the streams for this to work.

## PAD, CROP, AND SCALE VIDEOS

### 10.1 Pad

```
ffmpeg -i input.mp4 -vf "pad=width=1280:height=720:x=0:y=120:color=black" output.mp4
```

Will output to 1280x720 with the video 120 pixels from the top

### 10.2 Crop

```
ffmpeg -i in.mp4 -vf "crop=in_w:in_h" output.mp4
```

### 10.3 Scale

Documentation about scaling can be found [HERE](#).

#### 10.3.1 Scale to a Specific Size

```
ffmpeg -i input.mp4 -vf scale=1920:1080 output.mp4
```

#### 10.3.2 Downscale and Maintain Aspect Ratio

```
ffmpeg -i input.mp4 -vf scale="trunc(oh*a/2)*2:720" output.mp4
```

#### 10.3.3 Scale and Pad

```
ffmpeg -i input.mp4 -vf "scale=iw*min(1280/iw\,720/ih):ih*min(1280/iw\,720/ih),  
↪pad=1280:720:(1280-iw)/2:(720-ih)/2" -c:v libx264 output.mp4
```



## EXTRACTING FRAMES AND STREAMS

Note about using `-fps_mode passthrough` : If using an older version of ffmpeg, you may need to use `-vsync 0` instead. This is telling FFmpeg to passthrough the original input framerate. By default this is set to auto, which could result in dropped or skipped frames that would cause inaccurate results when trying to extract specific frames.

### 11.1 Extract One Frame

```
ffmpeg -i input_file -vf "select=gte(n\,frame_number)" -vframes 1 output_file.png
```

### 11.2 Extract Multiple Frames

```
ffmpeg -i input_file -vf "select=eq(n\,frame_number1)+eq(n\,frame_number2)+eq(n\,frame_↪number3)" -fps_mode passthrough output_file_%03d.png
```

### 11.3 Extract All Keyframes

```
ffmpeg -i input_file -vf "select=eq(pict_type\,I)" -fps_mode passthrough output_file-↪%03d.png
```

### 11.4 Extract a Stream

If the video only has a single video and audio track, we can simply exclude the video track.

```
ffmpeg -i input.mkv -vn -c:a copy output.wav
```

For more complex videos, we can use `-map` to identify the exact audio stream that we want to extract. The following example extracts the primary audio track:

```
ffmpeg -i input.mkv -map a:0 -c:a copy output.wav
```





## IMAGE SEQUENCES

### 12.1 Loop a Single Frame

```
ffmpeg -framerate 24 -loop 1 -i input.png -t 10 -c:v libx264 -pix_fmt yuv420p output.mp4
```

This can be tested using the image below:



FFmpeg

## 12.2 Image Sequence to Video

Method 1:

```
ffmpeg -framerate 24 -pattern_type glob -i *.png -c:v libx264 -pix_fmt yuv420p output.mp4
```

Method 2:

```
ffmpeg -framerate 24 -i %06d.png -c:v libx264 -pix_fmt yuv420p output.mp4
```

## USING FFPROBE

### 13.1 General Syntax

```
ffprobe -show_streams -show_format input_file
```

### 13.2 Specifying Fields

The following example uses the `-show_entries` command to specify a subset of stream fields that we want to look at. Note that `-loglevel error` is used to reduce the verbosity of FFprobe's output so it is easier to read. `-of default=noprint_wrappers=1` is also used to remove some of the formatting from the output and make it easier to read.

```
ffprobe -loglevel error -show_entries stream=color_range,color_space,color_transfer,  
↪color_primaries -of default=noprint_wrappers=1 input_file
```



## FRAME AND STREAM MD5S

FFmpeg has the ability to create MD5s of streams and frames. This can be especially helpful when attempting to determine if a transcode or stream copy is reversible/lossless. It also allows us to examine a stream's integrity at a granular level and better understand how different encodings of the same file may differ.

### 14.1 FrameMD5

Documentation about framemd5s can be found [HERE](#).

```
ffmpeg -i input_file.mov -f framemd5 output_file.framemd5
```

### 14.2 Stream MD5

Using the `-map` command to select the primary video track:

```
ffmpeg -loglevel quiet -i input.mp4 -map 0:v -f md5 -
```

Individually excluding other streams:

```
ffmpeg -loglevel quiet -i input.mp4 -an -dn -sn -f md5 -
```



## SPECTROGRAMS

### 15.1 Using FFmpeg

Documentation on FFmpeg's spectrogram filter can be found [HERE](#).

#### 15.1.1 Using FFplay

The following command creates a 1920x1080 spectrogram that scrolls as the audio plays.

```
ffplay -f lavfi "amovie='input.wav', asplit [a][out1]; [a]↵  
↵showspectrum=mode=separate:color=rainbow:slide=scroll:scale=lin:size=1920x1080 [out0]"
```

#### 15.1.2 Outputting to a File

For mono audio:

```
ffmpeg -i input.wav -lavfi [a:0]showspectrumpic output.png
```

For multi-channel audio:

```
ffmpeg -i input.wav -lavfi [a:0]showspectrumpic=mode=separate output.png
```

Changing the Orientation:

```
ffmpeg -i input.wav -lavfi showspectrumpic=mode=separate:orientation=1 output.png
```

### 15.2 Using SoX

If installed, SoX can be used as an alternative to FFmpeg for generating spectrograms.

```
sox input.wav -n spectrogram -o output.png
```





## INDICES AND TABLES

- `genindex`
- `modindex`
- `search`