Multimedia on Linux Command Line: wget, PdfTK, ffmpeg, flac, SoX

By Steven Gordon on Wed, 23/07/2014 - 5:33pm

Some examples of using the command line in Linux to:

- Modify PDF files, e.g. combine two files into one, re-order pages. Software: PdfTk [3]
- Convert video and audio files into different formats. Software: FFmpeg [4]
- Modify audio and video files, e.g. split files, extra an audio stream from a movie. Software: FFmpeg, flac [5]
- Record a screencast, i.e. audio from microphone and video of screen. Software: SoX [6], FFmpeg

Assumes basic command line knowledge, and some Bash scripting [7].

1. PDF

Download some example PDF files:

```
student@netlab01:~$ wget http://ict.siit.tu.ac.th/~sgordon/slides/bitcoin.pdf
--2014-07-23 13:46:22--  http://ict.siit.tu.ac.th/~sgordon/slides/bitcoin.pdf
Resolving ict.siit.tu.ac.th (ict.siit.tu.ac.th)... 203.131.209.82
Connecting to ict.siit.tu.ac.th (ict.siit.tu.ac.th)|203.131.209.82|:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 340296 (332K) [application/pdf]
Saving to: `bitcoin.pdf'

100%[===========================================================>] 340,296     1.80M/s   in 0.2s

2014-07-23 13:46:22 (1.80 MB/s) - `bitcoin.pdf' saved [340296/340296]

student@netlab01:~$ wget http://ict.siit.tu.ac.th/~sgordon/slides/internet-privacy-options.pdf
[...]
2014-07-23 13:47:09 (1.52 MB/s) - `internet-privacy-options.pdf' saved [706109/706109]

student@netlab01:~$ ls -l *.pdf
-rw-rw-r-- 1 student student 340296 Jul  2 16:09 bitcoin.pdf
-rw-rw-r-- 1 student student 706109 Jun 19 08:21 internet-privacy-options.pdf
```

Concatenate two PDF files (with `pdftk`) and view in PDF viewer (with `evince`):

```
student@netlab01:~$ pdftk internet-privacy-options.pdf bitcoin.pdf cat output privacy-and-bitcoin.pdf
student@netlab01:~$ ls -l *.pdf
-rw-rw-r-- 1 student student  340296 Jul  2 16:09 bitcoin.pdf
-rw-rw-r-- 1 student student  706109 Jun 19 08:21 internet-privacy-options.pdf
-rw-rw-r-- 1 student student 1452995 Jul 23 13:49 privacy-and-bitcoin.pdf
student@netlab01:~$ evince privacy-and-bitcoin.pdf
[view on GUI]
```

Select specific pages of PDF for rotation and joining. The command uses pages 1-2 of file A, rotated 90 degress to the West and pages 1-2 of file B, rotated 90 degress to the East. More recent versions of `pdftk` may use 'east' instead of 'E' etc. See the man page for syntax on your computer.

```
student@netlab01:~$ pdftk A=internet-privacy-options.pdf B=bitcoin.pdf cat A1-2W B1-2E output titles.pdf
```

Shuffle pages, i.e. 1 page from file A, then 1 page from file B, then next page from file A and so on:

```
student@netlab01:~$ pdftk A=internet-privacy-options.pdf B=bitcoin.pdf shuffle A B output shuffled.pdf
```

Burst a single PDF file into multiple files, 1 per page:

```
student@netlab01:~$ pdftk bitcoin.pdf burst output bitcoin-single_%02d.pdf
```

```
student@netlab01:~$ ls -l bitcoin*.pdf
-rw-rw-r-- 1 student student 340296 Jul  2 16:09 bitcoin.pdf
-rw-rw-r-- 1 student student  51609 Jul 23 14:05 bitcoin-single_01.pdf
-rw-rw-r-- 1 student student  52358 Jul 23 14:05 bitcoin-single_02.pdf
-rw-rw-r-- 1 student student  86916 Jul 23 14:05 bitcoin-single_03.pdf
[...]
-rw-rw-r-- 1 student student  59713 Jul 23 14:05 bitcoin-single_41.pdf
-rw-rw-r-- 1 student student  51691 Jul 23 14:05 bitcoin-single_42.pdf
```

Dump data about the PDF file:

```
student@netlab01:~$ pdftk bitcoin.pdf dump_data output bitcoin.txt
student@netlab01:~$ cat bitcoin.txt
InfoKey: Author
InfoValue: Networking
InfoKey: Producer
InfoValue: pdfTeX-1.40.14
InfoKey: Creator
InfoValue: LaTeX with Beamer class version 3.24
InfoKey: ModDate
InfoValue: D:20140702160659+07'00'
InfoKey: PTEX.Fullbanner
InfoValue: This is pdfTeX, Version 3.1415926-2.5-1.40.14 (TeX Live 2013/Debian) kpathsea version 6.1.1
InfoKey: Title
InfoValue: Bitcoin
InfoKey: CreationDate
InfoValue: D:20140702160659+07'00'
PdfID0: b773e8cb2645c51857f27f8d646a1997
PdfID1: b773e8cb2645c51857f27f8d646a1997
NumberOfPages: 42
BookmarkTitle: Cryptography Principles
[...]
PageLabelPrefix: 42
PageLabelNumStyle: NoNumber
```

Add a watermark to a PDF. First create a PDF that contains the watermark by using `groff` to produce a PostScript file, and `ps2pdf` to convert that PostScript file to PDF. Then use `pdftk` to stamp that background PDF to the original. A better way would be to create the background PDF using other tools (e.g. OpenOffice and then export to PDF) so it would match the size of the original PDF.

```
student@netlab01:~$ echo "Steven Gordon created this" | groff -Tps | ps2pdf - bg.pdf
student@netlab01:~$ pdftk bitcoin.pdf stamp bg.pdf output bitcoin-bg.pdf
```

2. Video and Audio Conversions

Default Ubuntu installs use avconv [8], which was a fork of the popular ffmpeg [4]. Some other Linux systems use ffmpeg instead. Personally I use ffmpeg (as that is what I have experience with). Therefore I have installed ffmpeg on the lab computers by compiling from source. If you want ffmpeg on your computer then the instructions for compiling [9] are very good. Give it a try.

On the lab computers you will find a `ffmpeg_build` and `ffmpeg_sources` directory which were used for compilation, and the executable files are in the `bin` directory. If you haven't already done so from a previous task, tour PATH needs to be set to include the `bin` directory, e.g.:

```
student@netlab01:~$ PATH=/home/student/bin:$PATH
```

If you have avconv, then most of the corresponding commands can be substitute as below, although sometimes there are different syntax for options between avconv and ffmpeg.

- ffmpeg -- avconv
- ffprobe -- avprobe
- ffplay -- avplay

2.1 View Media Information

Download the Tears of Steel [10], an open movie using open source software, especially Blender [11] for animations. Either download from the Tears of Steel website [10] or if in the lab, from the local server as below:

```
student@netlab01:~$ wget http://10.10.6.210/tears_of_steel_720p.mkv
```

View details of a media file:

```
student@netlab01:~$ ffprobe tears_of_steel_720p.mkv
ffprobe version 2.2.git Copyright (c) 2007-2014 the FFmpeg developers
  built on Jul 23 2014 19:00:41 with gcc 4.6 (Ubuntu/Linaro 4.6.3-1ubuntu5)
  configuration: --prefix=/home/student/ffmpeg_build --extra-cflags=-I/home/student/ffmpeg_build/include --e
  libavutil      52. 92.101 / 52. 92.101
  libavcodec     55. 69.100 / 55. 69.100
  libavformat    55. 48.101 / 55. 48.101
  libavdevice    55. 13.102 / 55. 13.102
  libavfilter     4. 11.102 /  4. 11.102
  libswscale      2.  6.100 /  2.  6.100
  libswresample   0. 19.100 /  0. 19.100
  libpostproc    52.  3.100 / 52.  3.100
Input #0, matroska,webm, from 'tears_of_steel_720p.mkv':
  Metadata:
    encoder         : libmkv 0.6.5
    TITLE           :
    ARTIST          :
    COMPOSER        :
    SYNOPSIS        :
    DATE_RELEASED   :
    GENRE           :
  Duration: 00:12:14.17, start: 0.000000, bitrate: 4167 kb/s
    Stream #0:0(eng): Video: h264 (Main), yuv420p(tv, bt709), 1280x534 [SAR 1:1 DAR 640:267], 24 fps, 24 tbr
    Stream #0:1(eng): Audio: aac, 44100 Hz, stereo, fltp (default)
```

The first lines are just the banner from ffmpeg (you can hide this using the `-hide_banner` option; I will not show it in subsequent output).
The last two lines show the media streams: video using H.264 codec and audio using AAC codec.

2.2 Split/Cut Videos

A quick way to extract a portion of video from a file (e.g. the first 5 minutes). Here we will select just 20 seconds of video, starting from 50 seconds into it, i.e. from 0m50s to 1m10s.

```
student@netlab01:~$ ffmpeg -ss 00:00:50 -i tears_of_steel_720p.mkv -t 00:00:20 -vcodec copy -acodec copy tos
Input #0, matroska,webm, from 'tears_of_steel_720p.mkv':
  Metadata:
    encoder         : libmkv 0.6.5
    TITLE           :
    ARTIST          :
    COMPOSER        :
    SYNOPSIS        :
    DATE_RELEASED   :
    GENRE           :
  Duration: 00:12:14.17, start: 0.000000, bitrate: 4167 kb/s
    Stream #0:0(eng): Video: h264 (Main), yuv420p(tv, bt709), 1280x534 [SAR 1:1 DAR 640:267], 24 fps, 24 tbr
    Stream #0:1(eng): Audio: aac, 44100 Hz, stereo, fltp (default)
Output #0, matroska, to 'tos.mkv':
  Metadata:
    GENRE           :
    TITLE           :
    ARTIST          :
    COMPOSER        :
    SYNOPSIS        :
    DATE_RELEASED   :
    encoder         : Lavf55.43.100
    Stream #0:0(eng): Video: h264 (H264 / 0x34363248), yuv420p, 1280x534 [SAR 1:1 DAR 640:267], q=2-31, 24 f
    Stream #0:1(eng): Audio: aac ([255][0][0][0] / 0x00FF), 44100 Hz, stereo (default)
Stream mapping:
  Stream #0:0 -> #0:0 (copy)
  Stream #0:1 -> #0:1 (copy)
Press [q] to stop, [?] for help
frame=  487 fps=0.0 q=-1.0 Lsize=    6750kB time=00:00:20.00 bitrate=2763.9kbits/s
video:6347kB audio:392kB subtitle:0kB other streams:0kB global headers:0kB muxing overhead: 0.154705%
```

The options are as follows (the ordering for some is important):

- `-ss 00:00:50` start time at 50 seconds
- `-i tears_of_steel_720p.mkv` is the input file
- `-t 00:00:20` is duration to copy
- `-vcodec copy` copies the video codec from input to output (i.e. does not re-code)
- `-acodec copy` copies the audio codec from input to output (i.e. does not re-code)
- `tos.mkv` is the output filename

2.3 Convert Videos

Converting videos (and audio files) can be quite complex as you need to consider the container format (e.g. MKV), video codec (H.264) and audio codec (AAC) and may convert one or all of these to another format/codec. Some formats/codecs require specific parameters (e.g. bit rates, channels, resolutions). First lets see what formats (containers) and codecs are supported. avconv supports reading of some formats/codecs (demux, decode) and writing to others (mux, encode).

```
student@netlab01:~$ ffmpeg -formats
File formats:
 D. = Demuxing supported
 .E = Muxing supported
 --
  E 3g2             3GP2 (3GPP2 file format)
  E 3gp             3GP (3GPP file format)
 D  4xm             4X Technologies
  E a64             a64 - video for Commodore 64
 D  aac             raw ADTS AAC (Advanced Audio Coding)
 DE ac3             raw AC-3
[...]
 D  xmv             Microsoft XMV
 D  xwma            Microsoft xWMA
 D  yop             Psygnosis YOP
 DE yuv4mpegpipe    YUV4MPEG pipe
student@netlab01:~$ ./ffmpeg -codecs
Codecs:
 D..... = Decoding supported
 .E.... = Encoding supported
 ..V... = Video codec
 ..A... = Audio codec
 ..S... = Subtitle codec
 ...I.. = Intra frame-only codec
 ....L. = Lossy compression
 .....S = Lossless compression
 -------
 D.VI.. 012v                 Uncompressed 4:2:2 10-bit
 D.V.L. 4xm                  4X Movie
 D.VI.S 8bps                 QuickTime 8BPS video
[...]
 D.S... text                 raw UTF-8 text
 D.S... vplayer              VPlayer subtitle
 DES... webvtt               WebVTT subtitle
 DES... xsub                 XSUB
```

Now lets try some conversions on the 20 second clip.

```
student@netlab01:~$ ffmpeg -i tos.mkv tos.mp4
student@netlab01:~$ ffmpeg -i tos.mkv tos.webm
student@netlab01:~$ ffmpeg -i tos.mkv tos.avi
student@netlab01:~$ ls -lh tos.*
-rw-rw-r-- 1 student student 1.9M Jul 23 16:13 tos.avi
-rw-rw-r-- 1 student student 6.6M Jul 23 16:01 tos.mkv
-rw-rw-r-- 1 student student 3.0M Jul 23 16:09 tos.mp4
-rw-rw-r-- 1 student student 732K Jul 23 16:10 tos.webm
```

Extension	*Container*	*Video*	*Audio*
mp4	MP4	H.264	AAC
mkv	Matroska	H.264	AAC
webm	WebM	VP8	Ogg Vorbis
avi	AVI	MPEG-4	MP3

The difficult part of video conversions is selecting the correct parameters. Not just codecs, but also bit rates, resolution, channels, sampling rates and codec specific parameters. ffmpeg will choose reasonable default parameters some times, but other times you will need to specify them yourself. Here are some examples that convert again to WebM. The first creates a low quality output, reducing the resolution to 640x267 (compared to the original 1280x534). The second creates a high quality output, setting the quality factor to 20 (which results in an average bitrate of 578kb/s, compared to the original 294kb/s). Is there any visual difference in the output of the videos?

```
student@netlab01:~$ ffmpeg -i tos.mkv -vcodec libvpx -s 640x267 -acodec libvorbis -ab 64k -f webm tos-low.we
student@netlab01:~$ ffmpeg -i tos.mkv -vcodec libvpx -q 20 -acodec libvorbis -f webm tos-hi.webm
student@netlab01:~$ ffprobe -hide_banner tos.webm
Input #0, matroska,webm, from 'tos.webm':
  Metadata:
    title           :
    encoder         : Lavf55.43.100
  Duration: 00:00:20.33, start: 0.000000, bitrate: 294 kb/s
    Stream #0:0(eng): Video: vp8, yuv420p, 1280x534, SAR 1:1 DAR 640:267, 24 fps, 24 tbr, 1k tbn, 1k tbc (de
    Stream #0:1(eng): Audio: vorbis, 44100 Hz, stereo, fltp (default)
student@netlab01:~$ ffprobe -hide_banner tos-low.webm
Input #0, matroska,webm, from 'tos-low.webm':
  Metadata:
    title           :
    encoder         : Lavf55.43.100
  Duration: 00:00:20.33, start: 0.000000, bitrate: 200 kb/s
    Stream #0:0(eng): Video: vp8, yuv420p, 640x267, SAR 1:1 DAR 640:267, 24 fps, 24 tbr, 1k tbn, 1k tbc (def
    Stream #0:1(eng): Audio: vorbis, 44100 Hz, stereo, fltp (default)
student@netlab01:~$ ffprobe -hide_banner tos-hi.webm
Input #0, matroska,webm, from 'tos-hi.webm':
  Metadata:
    title           :
    encoder         : Lavf55.43.100
  Duration: 00:00:20.33, start: 0.000000, bitrate: 578 kb/s
    Stream #0:0(eng): Video: vp8, yuv420p, 1280x534, SAR 1:1 DAR 640:267, 24 fps, 24 tbr, 1k tbn, 1k tbc (de
    Stream #0:1(eng): Audio: vorbis, 44100 Hz, stereo, fltp (default)
```

To select the correct parameters you often need to know about the specific codecs. Read the ffmpeg man page for the many options available.

2.4 Extract Audio Stream from a Video

Movies usually have at least two stream: video and audio (some may have multiple audio streams, e.g. in different languages or codecs). You can separate them, for example saving just the audio from a movie to a file. The first command below copies the audio stream, while the second also converts it to 64kb/s/s MP3.

```
student@netlab01:~$ ffmpeg -i tos.mkv -vn -acodec copy tos-audio.aac
student@netlab01:~$ ffmpeg -i tos.mkv -vn -acodec mp3 -ab 64k tos-audio-lo.mp3
```

2.5 Convert Audio Files

We can convert audio in the same manner as video. It is usually a bit easier to understand as there is only a single codec to be concerned with.

```
student@netlab01:~$ ffmpeg -i tos-audio.aac tos-audio.mp3
student@netlab01:~$ ffmpeg -i tos-audio.aac tos-audio.wma
student@netlab01:~$ ffmpeg -i tos-audio.aac tos-audio.flac
```

2.6 FLAC Audio

The Free Audio Lossless Codec (FLAC) is a good choice for encoding audio if you want to maintain the quality from the original source. You can process FLAC in ffmpeg in the same way as other audio codecs. However there are also other tools, one called simply `flac`. Lets try some operations on the soundtrack of Tears of Steel (the stereo version in FLAC format is available via Xiph.org [12]).

The file info:

```
student@netlab01:~$ ffprobe -hide_banner tearsofsteel-stereo.flac
Input #0, flac, from 'tearsofsteel-stereo.flac':
  Metadata:
    TITLE           : Tears of Steel
    ARTIST          : Blender Foundation
    DATE            : 2012
    COPYRIGHT       : (CC) Blender Foundation | mango.blender.org
    LICENSE         : http://creativecommons.org/licenses/by/3.0/
    comment         : DVD stereo mix
  Duration: 00:12:14.00, start: 0.000000, bitrate: 913 kb/s
    Stream #0:0: Audio: flac, 48000 Hz, stereo, s32
student@netlab01:~$
student@netlab01:~$
```

Now use `flac` to decode to obtain a PCM WAVE output. At the same time we will cut, only getting from time 50s to 1m10s.

```
student@netlab01:~$ flac -d tearsofsteel-stereo.flac --skip=00:50.00 --until=01:10.00
flac 1.2.1, Copyright (C) 2000,2001,2002,2003,2004,2005,2006,2007  Josh Coalson
flac comes with ABSOLUTELY NO WARRANTY.  This is free software, and you are
welcome to redistribute it under certain conditions.  Type `flac' for details.

tearsofsteel-stereo.flac: done
student@netlab01:~$ ls -l tearsofsteel-stereo.*
-rw-rw-r-- 1 student student 83769154 Sep 29  2012 tearsofsteel-stereo.flac
-rw-rw-r-- 1 student student  5760044 Sep 29  2012 tearsofsteel-stereo.wav
```

3. Sreencasts

During my lectures I record my screen as well as audio from microphone, and then post the resulting video screencast on Youtube. Lets see how.

3.1 Recording Audio from Microphone

Many applications, including ffmpeg, can be used to record audio. I use SoX [6], specifically the `rec` command (which is just a shortcut to `sox` but by default selects the default input device, typically the microphone). Here is an example that records single channel 16-bit audio at 44.1kHz, saving in FLAC format.

```
student@netlab01:~$ rec -r 44100 -b 16 -c 1 audio.flac

```

3.2 Recording the Screen

I use ffmpeg to capture the screen as follows, with the options explained below:

```
student@netlab01:~$ ffmpeg -f x11grab -r 10 -s 1600x900 -i :0.0+0,0 -vcodec libx264 -preset ultrafast -crf 0
```

- `-f x11grab` indicates to take the input from the X11 display (i.e. the screen on Linux)
- `-r 10` is the frame rate (generally in my lectures there is little motion, so a low frame rate is ok)
- `-s 1600x900` is the resolution of the screen
- `-i :0.0` indicates the display number (should not need to change this)
- `+0,0` is the offset from the top-left of the screen to record (this will be used later)
- `-vcodec libx264` specifics to use x264 to encode to H.264 codec
- `-preset ultrafast -crf 0` are options for the x264 codec
- `screen.mp4` is the output file

3.3 Recording the Projector (when Multiple Monitors)

In lectures I have multiple monitors. My normal configuration is the laptop screen (resolution 1366x768) is the first display and the projector (resolution 1024x768) extends that display to the right. That is, from the point of view of applications (including ffmpeg), I have a single display is 2390x768.

```
+==================+===============+
|                  |               |
|                  |               |
|      laptop      |   projector   |
|     1366x768     |   1024x768    |
|                  |               |
|                  |               |
+==================+===============+

 <----- 1366 -------> <---- 1024 ---->
 <-------------- 2390 --------------->
```

When recording the lecture I want to just recorder the projector portion of the display, not that on my laptop. With ffmpeg you can specify an offset from the top-left coordinates which are (0,0) to which you record. In the above example command the size of the display to record was set (using the `-s` option) to 1600x900. This needs to be changed to the size of the projector, 1024x768. And in the above example we use a 0 offset, i.e. `+0,0`. Here we need to shift to the right by the size of the laptop screen, i.e. `+1366,0`. Then only the projector portion of the display will be recorded. The resulting command is:

```
student@netlab01:~$ ffmpeg -f x11grab -r 10 -s 1024x768 -i :0.0+1366,0 -vcodec libx264 -preset ultrafast -cr
```

You can see the size and offset to record just a portion of the display.

3.4 Starting Audio and Screen Record

I use a script to start the audio and screen recording at the same time. A cut down version is shown below. The full script I use, which includes some other features, can be downloaded [13]. Put it into your `bin` directory (which should be in the PATH).

```
#!/bin/bash
rec -q -r 44100 -b 16 -c 2 audio.flac &
audioPID=$!
ffmpeg -f x11grab -r 10 -s 1024x768 -i :0.0+1366,0 -vcodec libx264 -preset ultrafast -crf 0 screen.mp4 &
screenPID=$!
op="s"
while [ ${op} != "z" ]
do
        read -n1 op
done
kill ${audioPID} ${screenPID}
exit
```

The full screencast [13] script be be run as:

```
student@netlab01:~$ screencast netlab test
```

3.5 Combing Audio and Screen Files

After using `screencast` to record the audio and screen into separate files, we need to combine (multiplex) them into a single file. I will do it in three steps.

First convert the FLAC audio to WAV:

```
student@netlab01:~$ flac -d test-audio.flac
```

Second convert the WAV audio to AAC:

```
student@netlab01:~$ faac -b 64 test-audio.wav
```

Finally combine the audio (AAC) and screen (MP4) to produce an MP4 video:

```
student@netlab01:~$ ffmpeg -i test-screen.mp4 -i test-audio.aac -absf aac_adtstoasc -vcodec copy -acodec cop
```

For my lectures I have a script called audioscreen2video [14] that combines the above commands as well as a few others to create the video. Run it as:

```
student@netlab01:~$ audioscreen2video test 2014-07-25
```

Content: Howto [15]
Interest: ffmpeg [16]
sox [17]
vlc [18]
Linux [19]

Source URL: http://sandilands.info/sgordon/multimedia-on-linux-command-line

Links:
[1] http://sandilands.info/sgordon/multimedia-on-linux-command-line
[2] http://sandilands.info/sgordon/user/2
[3] http://www.pdflabs.com/tools/pdftk-the-pdf-toolkit/
[4] https://www.ffmpeg.org/
[5] https://xiph.org/flac/
[6] http://sox.sourceforge.net/
[7] http://sandilands.info/sgordon/aliases-prompts-and-scripting-in-linux
[8] https://libav.org/
[9] https://trac.ffmpeg.org/wiki/CompilationGuide/Ubuntu
[10] http://tearsofsteel.org/
[11] http://www.blender.org/
[12] http://media.xiph.org/tearsofsteel/
[13] http://sandilands.info/sgordon/doc/code/screencast
[14] http://sandilands.info/sgordon/doc/code/audioscreen2video
[15] http://sandilands.info/sgordon/taxonomy/term/212
[16] http://sandilands.info/sgordon/taxonomy/term/322
[17] http://sandilands.info/sgordon/taxonomy/term/324
[18] http://sandilands.info/sgordon/taxonomy/term/304
[19] http://sandilands.info/sgordon/taxonomy/term/300