



Kyoshi Tagahase Segundo, 123RF

Un script Perl para empalmar vídeos

PERL FILMS PRESENTA...

Un vídeo instructivo parece mucho más profesional si añadimos créditos de apertura. Las herramientas Memcoder y Sox nos ayudan a manejar formatos caprichosos, y un script Perl automatiza el proceso.

POR MICHAEL SCHILLI

YouTube nos ofrece un fascinante número de vídeos con instrucciones que cubren diferentes temáticas. Ya sea que queramos ver chefs amateur cocinando sus platos favoritos, manitas demostrando sus habilidades abriendo candados, o dueños de coches inclinados por completo reparando sus propios vehículos, YouTube casi siempre tiene un vídeo que cubre lo que buscamos.

Tras grabarlo, seguramente querremos añadir créditos de apertura. Los videoaficionados amateur siempre se pueden permitir dedicar un par de segundos a informar a la audiencia un poco más acerca del vídeo que se va a mostrar. Para ello, podríamos usar algunos programas propietarios bajo Windows como Adobe Pre-

miere, o algún programa de Mac como iMovie o Final Cut, o incluso podríamos intentarlo con alguna aplicación para Linux como Cinelarra. Sin embargo, en lugar de usar estos mastodontes, vamos a explicar un método rápido y limpio basado en línea de comandos con la ayuda de un pequeño script Perl que confía en dos aplicaciones externas de vídeo y sonido: *sox* y *memcoder*.

Fotogramas

Los vídeos se componen de imágenes individuales reproducidas en una rápida sucesión conocida como fotogramas. Las videocámaras normales capturan unos 30 fotogramas por segundo, y un programa como *mplayer* reproducirá los fotogramas

a una tasa fija. Un vídeo estático compuesto por unos títulos puede convertirse fácilmente desde una serie de imágenes JPG idénticas a un archivo AVI con el uso de Memcoder. Si a continuación concatenamos los dos archivos de vídeo, obtenemos una película con títulos de apertura, al menos en teoría. En la vida real, podemos encontrar un par de problemas.

¿Qué es un Códec?

Los archivos multimedia en formato AVI actúan como contenedores de flujos de vídeo y audio, los cuales se reproducen de manera simultánea por un reproductor de vídeo. Tanto el vídeo como el audio del contenedor AVI se pueden guardar en variedad de formatos. La pista de audio utiliza generalmente el formato PCM, con información prácticamente en bruto, o puede estar comprimida en forma de archivo MP3.

La información de vídeo necesita comparativamente una cantidad de memoria mucho mayor, como podemos imaginar fácilmente, dado que tenemos 30 archi-

vos de imagen por cada segundo de vídeo. Esto también explica el porqué el método de codificación, o códec, es tan crucial: un buen códec puede comprimir información de manera muy importante sin comprometer la calidad de la imagen. Entre todos los codecs disponibles, muchos están patentados.

Aunque un contenedor AVI puede albergar toda una variedad de información de vídeo y audio codificados, no podemos cambiar sin más el método de codificación en mitad del flujo. En otras palabras, para concatenar los créditos y un archivo de vídeo, tenemos que asegurarnos de que ambos usan el mismo códec desde el inicio, o al menos necesitamos usar una herramienta como *men-coder* para convertir las diferentes codificaciones al mismo formato de salida.

Comparación de Cámaras

La Figura 1 es un listado de la metainformación que se ha analizado a partir de

```

$ meta coolpix.avi
{
  audio_bitrate => 64_000,
  audio_codec   => "pcm",
  audio_format  => 1,
  audio_id      => 1,
  audio_nch     => 1,
  audio_rate    => 8000,
  clip_info_n   => 1,
  clip_info_name0 => "Software",
  clip_info_value0 => "",
  demuxer       => "avi",
  filename      => "coolpix.avi",
  "length"      => "29.00",
  video_aspect  => "0.0000",
  video_bitrate => 10_669_112,
  video_codec   => "ffmjpeg",
  video_format  => "MJPG",
  video_fps     => "30.000",
  video_height  => 480,
  video_id     => 0,
  video_width  => 640,
}

$ meta camcorder.avi
{
  audio_bitrate => 1_024_000,
  audio_codec   => "pcm",
  audio_format  => 1,
  audio_id      => 1,
  audio_nch     => 2,
  audio_rate    => 32_000,
  demuxer       => "avini",
  filename      => "camcorder.avi",
  "length"      => "6.01",
  video_aspect  => "0.0000",
  video_bitrate => 28_771_224,
  video_codec   => "ffdv",
  video_format  => "dvsd",
  video_fps     => "29.970",
  video_height  => 480,
  video_id     => 0,
  video_width  => 720,
}

```

Figura 1: Metadatos de dos vídeos. En la parte superior, la Nikon Coolpix S52. Abajo, la Canon Elura 100.

dos archivos de vídeo por el programa del Listado 1. El programa usa el método *meta()* del módulo *Video::FrameGrab* de CPAN para recuperar las características del archivo de vídeo, y guardarlas a continuación en un hash.

La Figura 1 compara la metainformación de dos vídeos, *coolpix.avi* y *camcorder.avi*. El primer vídeo se ha realizado con una pequeña cámara de fotos de bolsillo, una Nikon Coolpix S52, y el segundo con una

cámara de vídeo digital de Canon, una Elura 100. Ambas cámaras tomaron los vídeos a una tasa de unos 30 fotogramas por segundo (*video_fps*), pero la cámara de Canon ha utilizado el códec *ffdv* (véase el campo *video_codec*), mientras que la Nikon ha utilizado *ffmjpeg*.

Las dos cámaras también usan diferentes formatos para guardar la información de audio. Mientras la cámara de vídeo ha usado dos canales (para el estéreo: el número de canales en *audio_nch* es 2), la Nikon sólo soporta mono (*audio_nch* es 1). La calidad de la audio es también diferente, ya que la cámara de vídeo usa una tasa de muestreo de 32.000 muestras por segundo para la grabación (campo *audio_rate*), comparados con las 8.000 muestras por segundo de la Nikon.

La Figura 1 también señala que la Nikon, con un *audio_rate* de 8000, tiene un *audio_bitrate* de 64000 (que es el requerimiento de memoria total en bits por segundo). Cada muestra por tanto tiene 8 bits, lo que da como resultado el denominado "tamaño de muestra" de exactamente 1 byte. La cámara de vídeo, en comparación, usa una tasa de muestreo de 32 bits (1.024.000 dividido entre 32.000), lo que da como resultado 16 bits por canal y un tamaño de muestra de 2 bytes.

Como puede ver de toda esta información, estos silenciosos créditos de apertura no pueden fusionarse en una grabación de vídeo de una cámara desconocida sin tener que hacer algún tipo de conversión. Afortunadamente, las herramientas

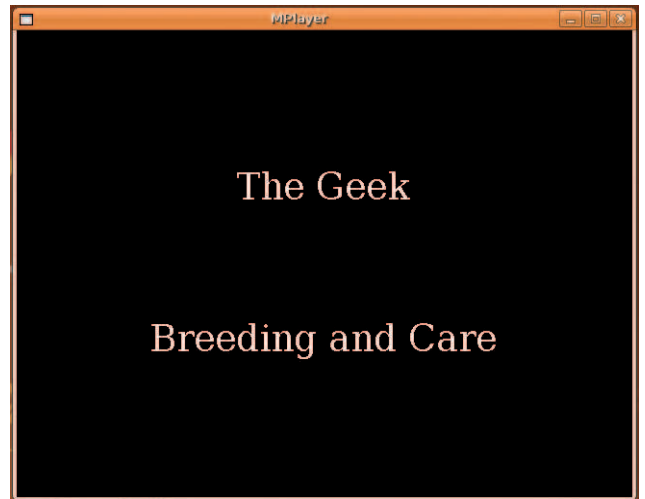


Figura 2: Créditos de apertura realizados mediante script, ejecutándose con MPlayer antes de la función principal.

Memcoder y Sox proporcionan las funciones que necesitamos para modificar los diferentes formatos y por tanto permitir que los créditos y el vídeo coexistan pacíficamente en el contenedor AVI.

Mencoder para Todos

Los usuarios que se enfrentan a los comandos de Mencoder por primera vez generalmente se asustan de su aparente complejidad. Incluso la más simple de las funciones parece requerir una combinación totalmente absurda de opciones. Una inspección más detenida revela que Mencoder no es tan difícil de utilizar: Para convertir un archivo de vídeo a otro formato, Mencoder aguarda el primer archivo como primer argumento, seguido por la acción de conversión, seguida del parámetro *-o* que configura el archivo de salida, como vemos en:

Listado 1: video-meta

```

01 #!/usr/local/bin/perl -w
02 use strict;
03 use Data::Dump qw(dump);
04 use Video::FrameGrab;
05
06 my($file) = @ARGV;
07 die "usage: $0 file" unless
    defined $file;
08
09 my $grabber =
    Video::FrameGrab->new(
10   video => $file);
11
12 my $meta =
    $grabber->meta_data();
13 print dump($meta), "\n";

```

```
mencoder input.avi [options]
-o output.avi
```

De igual manera, también podemos crear fácilmente un único archivo de salida a partir de múltiples archivos de entrada introduciendo los nombres de archivo uno tras otro en línea de comandos en lugar de *input.avi* (*input1.avi*, *input2.avi*,...).

Existen dos grupos de opciones de conversión para los componentes de vídeo y audio. Para pasar el flujo de audio desde el archivo de entrada al archivo de salida sin ninguna modificación, simplemente escribimos *-oac copy* (*a* de audio). Si queremos volver a codificar la pista de audio, teclearíamos *-oac pcm* para el formato PCM (Pulse Code Modulation) o *-oac mp3lame* para un formato MP3 creado

por el codificador de MP3 Lame. Si el codificador que usamos (Lame en nuestro ejemplo) también necesita algunas opciones, como *vbr 3*, simplemente las añadimos a la línea de comandos de Mencoder con la ayuda de la opción *-lameopts*:

```
-oac mp3lame -lameopts vbr=3
```

Para el componente de vídeo de un archivo AVI se usa un método similar. Para copiar el formato de vídeo sin cambios usamos *-ovc copy* (con *v* de vídeo). Para recodificar el formato de vídeo al formato MJPEG y pasarle la opción *vcoddec=mjpeg* al codificador, tecleamos *-ovc lavc -lavcopts vcoddec=mjpeg* en la línea de comandos de Mencoder. Armados con este conocimiento, deberíamos ser capa-

ces de transformar cualquier vídeo de un formato a otro sin muchos problemas.

Automatización Perl

El generador de créditos en *video-title-add* (véase el Listado 2) aguarda tres parámetros: el archivo de vídeo en el cual vamos a añadir los créditos y dos cadenas que se usaron como la primera y segunda líneas en los créditos de apertura del vídeo. Si realizamos la llamada al generador con los parámetros

```
video-title-add testvideo.avi
"El Geek" "Cría y Cuidados"
```

se crea un nuevo archivo *.avi* denominado *testvideo-withtitle.avi* que nos ofrece dos segundos de créditos de aper-

Listado 2: video-title-add

```
001 #!/usr/local/bin/perl -w
002 use strict;
003 use Sysadm::Install qw(:all);
004 use Imager;
005 use Imager::Fill;
006 use Log::Log4perl qw(:easy);
007 use Video::FrameGrab;
008 use File::Temp qw(tempdir temp-
009   file);
010 sub shell;
011
012 my $title_length = 2; # length
013   in seconds
014 my $FONT_FILENAME =
015   "/usr/share/fonts/" .
016   "truetype/ttf-bitstream-vera
017   /VeraSe.ttf";
018 my($video_file, $upper,
019   $lower) = @ARGV;
020 die "usage: $0 ",
021   "video_file upper_text
022   lower_text"
023 unless defined $upper;
024
025 (my $video_out = $video_file)
026   =~
027   s/(\[^\.]++$)/-withtitle$1/;
028
029 my $throwaway_file =
030   "throwaway_file(". $video_file .
031   ".avi)";
032 my $audio_title =
033   "throwaway_file(". $audio_title .
034   ".wav)";
035 my $audio_total =
036   "throwaway_file(". $audio_total .
037   ".wav)";
038
039 my $grabber =
040   Video::FrameGrab->new(
041     video =>
042       $video_file);
043 my $meta =
044   $grabber->meta_data();
045 my $height =
046   $meta->{video_height};
047 my $width =
048   $meta->{video_width};
049 my $dir = jpeg_dir_create(
050   $width, $height, $upper,
051   $lower,
052   $meta->{video_fps} *
053   $title_length);
054 shell qw(mencoder -nosound),
055   "mf://$dir/*.jpg",
056   qw(-mf fps=30 -o),
057   $video_title,
058   qw(-ovc lavc -lavcopts
059     vcoddec=mjpeg);
060 my $sample_size =
061   $meta->{audio_bitrate} /
062   $meta->{audio_rate} /
063   $meta->{audio_nch} / 8;
064
065 silent_wav( $title_length,
066   $audio_title,
067   $meta->{audio_rate},
068   $meta->{audio_nch},
069   $sample_size);
070
071 shell qw(mplayer -vc null -vo
072   null -ao
073   pcm), $video_file;
074
075 shell "sox", $audio_title,
076   "audiodump.wav", "-o",
077   $audio_total;
078
079 shell "mencoder", "-nosound",
080   $video_title,
081   $video_file, qw(-ovc
082     lavc -lavcopts
083     vcoddec=mjpeg -o),
084   $video_mum;
085
086 # add sound
087 shell "mencoder", $video_mum,
088   qw(-oac copy
089     -audiofile), $audio_total,
090   qw(-ovc copy -o),
091   $video_out;
092
093 #####
094 sub throwaway_file {
095   #####
096 }
```

tura antes de la función principal, como se muestra en la Figura 2.

El script comienza llamando a la función `jpeg_dir_create`, definida en el Listado 2, empezando en la línea 95, que crea `$n` imágenes JPG idénticas con un ancho de `$w` y un alto de `$h` en un directorio temporal. Las imágenes muestran las líneas de texto pasadas como `$upper` y `$lower` sobre un fondo de pantalla negro. Según lo dicho, un vídeo de dos segundos con una tasa de 30 fotogramas por segundo requiere exactamente 60 imágenes. El programa principal por tanto fija `$n` a 60.

El script usa entonces el módulo `Imager` de CPAN para crear un nuevo objeto imagen `Imager` con las dimensiones de `$w` por `$h`. Define el color negro como un objeto de la clase `Imager::Color`, el cual

inicializa con un valor RGB de 0-0-0. La ruta guardada en la variable `$FONT_FILENAME` apunta al archivo TTF con el tipo de letra necesario y puede modificarse para reflejar nuestro entorno local según nuestras necesidades.

El método `align()` del objeto del tipo de letra toma la cadena de caracteres y la dibuja en una posición predefinida de la imagen. La directiva `center` alinea la cadena en torno al centro del eje `x`. La primera llamada a `align()` dibuja la línea `$upper` en torno a un tercio de la altura de la pantalla a partir de la parte superior. La segunda llamada dibuja `$lower` en los dos tercios de la altura de la pantalla. La imagen JPG, creada por las sucesivas llamadas a `write()`, se guarda entonces en un directorio temporal creado para

este propósito. El bucle `for` de la línea 128 añade 59 enlaces fuertes apuntando al archivo `c.jpg` que se acaba de crear. Esto engaña a `Mencoder` en la línea 43, que piensa que tiene 60 archivos en este directorio, aunque en realidad sólo ocupa el espacio de uno. El códec usado aquí es `mjpeg`, ya que es el que usa la pequeña cámara Nikon, y la calidad del vídeo conjunto sufrirá si se convierte en un formato de codificación con pérdidas a otro.

El Sonido del Silencio

Los créditos creados por `Mencoder` en la línea 43 no tienen hasta el momento banda sonora. No se ha asignado señal de audio a las imágenes JPG, y se le indica a `Mencoder` que deje de quejarse acerca de esto configurando la opción `-noaudio`. Desafortunadamente,

Listado 2: video-title-add

```

075 my($suffix) = @_;
076
077 my($fh, $file) = tempfile(
078     UNLINK => 1,
079     SUFFIX => $suffix,
080 );
081 return $file;
082 }
083
084 #####
085 sub shell {
086 #####
087 my($stdout, $stderr, $rc) =
088     tap@_;
089 if($rc) {
090     die "Command@_ failed:
091         $stderr";
092 }
093
094 #####
095 sub jpeg_dir_create {
096 #####
097 my($w, $h, $upper, $lower, $n)
098     = @_;
099 my $img = Imager->new(xsize =>
100     $width,
101     ysize => $height);
102 my $black =
103     Imager::Color->new( 0,0,0 );
104 $img->box(color=> $black,
105     filled=> 1);
106
107     105
108     106 my $font = Imager::Font->new(
109         file =>
110         $FONT_FILENAME) or die
111         Imager->errstr;
112
113     107 $font->align(string =>
114         $upper,
115         size => 38, color => "white",
116         x => $width/2, y => $height/3,
117         halign => "center", valign =>
118         "center",
119         image => $img );
120
121     108 $font->align(string =>
122         $lower,
123         size => 38, color => "white",
124         x => $width/2, y =>
125         $height*2/3,
126         halign => "center", valign =>
127         "center",
128         image => $img );
129
130     109 my($dir) = tempdir( CLEANUP =>
131         1 );
132 my $img_file = "$dir/c.jpg";
133 $img->write(file =>
134     $img_file) or
135     die "Cannot write ($!)";
136
137     110 for(1..$n-1) {
138         131 link $img_file, $link or die
139             $!;
140         132 cdback;
141         133 }
142         134
143         135 return $dir;
144         136 }
145         137
146         138 #####
147         139 sub silent_wav {
148         140 #####
149         141 my($secs, $outfile, $rate,
150             $channels,
151             $sample_size) = @_;
152         142
153         143 my($fh, $tempfile) =
154         144 tempfile( UNLINK => 1,
155             SUFFIX => ".dat" );
156         145
157         146 print $fh "SampleRate
158             $rate\n";
159         147 my $samples = $secs * $rate;
160         148
161         149 for( my $i = 0; ($i < $samples);
162             $i++) {
163         150
164         151 print $fh $i / $rate,
165             "\t0\n";
166         152 }
167         153 close $fh;
168         154
169         155 shell "sox", $tempfile, "-r",
170             $rate,
171             "-u", "-$sample_size",
172             "-c",
173             $channels, $outfile;
174         156
175         157 }
176         158
177         159 }

```

no se puede pegar un vídeo sin audio a otro con audio. Esto significa que se necesita un script para crear un archivo de sonido que contenga dos segundos de silencio.

La función `silent_wav()` que comienza en la línea 95 aguarda la longitud en segundos, el nombre del archivo resultante, la tasa `$rate`, el número de canales `$channels` y el tamaño de muestra `$sample_size` de la pista de audio con silencio. Se crea un nuevo archivo temporal con la extensión `.dat` y se guarda la información en bruto como bytes nulos (véase la Figura 3). La utilidad Sox toma este archivo en la línea 156 y lo convierte en el archivo WAV que necesitamos.

Receta Especial de Perlmeister

De vuelta al programa principal, todo lo que necesitamos en realidad es pegar la banda sonora a los créditos y luego pegar los archivos AVI. Desafortunadamente, Mencoder no puede hacer esto sin desincronizar las pistas de audio, dando como resultado problemas de sincronización inaceptables entre las pistas de audio y vídeo en la película resultante. Lo que podemos hacer, sin embargo, es extraer la pista de audio del vídeo original, pegarle la pista de audio con el silencio creado anteriormente, y fusionar la pista de audio completa con los dos vídeos sin audio que ya hemos fusionado previamente.

La llamada a `mplayer` de la línea 57 extrae la pista de audio del vídeo original en un archivo denominado `audiodump.wmv`. La línea 60 añade al comienzo la pista con el audio silencioso, que hemos guardado en el archivo `$audio_total`. La línea 63 inicia Mencoder, fusiona `$video_title` con `$video_file` con la opción `-nosound`, y convierte el resultado en un archivo AVI con un flujo de vídeo codificado en MJPEG.

Uno podría pensar que Mencoder sería capaz de añadir archivos de vídeo en

formato MJPEG creados a partir de imágenes JPEG a otro vídeo en formato MJPEG sin tener muchos problemas con el códec. Extrañamente, Mencoder finaliza mostrando un mensaje que señala que no está contento con la codificación que estamos usando. Sin embargo, si dejamos que Mencoder convierta el archivo de la cámara de vídeo a MJPEG, podemos añadirsele sin demasiado problema. En realidad es una pena, ya que recodificar un vídeo lleva más o menos el tiempo que dura la propia película, mientras que la opción `-ovc copy` pasa a toda velocidad por el formato mucho más rápidamente. ¡No tengo la culpa!

Sonido para el Vídeo Silencioso

Ahora, lo único que se debe hacer es añadir la pista de audio completa, `$audio_total`, al vídeo. El comando `mencoder` de la línea 68 usa la opción `-audiofile` para hacer justamente eso, y `-ovc copy` le indica que no toque la codificación de vídeo. El resultado `.avi` se escribe en el archivo definido por `$video_out` (es decir, en `testvideo-withtitle.avi`).

El script usa un par de funciones, algunas de las cuales se definen en mismo script y otras las toma del módulo `Sysadm::Install` de CPAN. Por ejemplo, la función `throwaway_file()` definida al principio de la línea 73 crea un archivo temporal con la extensión `$suffix`, lo cual es importante debido a que algunos programas usan la extensión del archivo para averiguar el formato de archivo. El módulo `File::Temp` de CPAN administra los archivos temporales, borrándolos cuando se completa el script.

La función `shell()` definida al principio de la línea 85 ejecuta un comando de shell que se le pasa en forma de lista, lo verifica para ver si funciona, y se queja si algo va mal. La declaración de la función en la línea 10 permite llamar a la función más tarde sin necesidad de paréntesis. La shell usa la función `tap()` del módulo `Sysadm::Install` de CPAN para llamar al programa externo, capturar la salida estándar y de errores, y retornarlos junto con el código devuelto.

Instalación

Las herramientas Mencoder, Mplayer y Sox a menudo están preinstaladas en sistemas Linux; si no lo están, podemos instalarlas en Debian por ejemplo, del siguiente modo:

```
sudo apt-get install sox 2
mencoder mplayer
```

Los módulos `Sysadm::Install`, `Log::Log4perl`, `Imager` e `Imager::Fill` también están disponibles como paquetes Debian. Si no es el caso de su distribución, una shell de CPAN nos ayudará con la instalación. De cualquier modo, necesitará usarla para instalar el módulo `Video::FrameGrab`. Adicionalmente, podría tener que modificar la ruta al archivo de tipos de letra True Type para `VeraSe.ttf`, como se define en la línea 13, para que coincida con nuestro entorno local.

Créditos Finales

Además de los créditos de apertura, un tráiler también puede mejorar mucho el valor de un vídeo. Para añadir uno, solamente tenemos que modificar el script para que cree una segunda película silenciosa para el tráiler, le parcheamos una pista de sonido silenciosa, `$audio_trailer` (o simplemente usamos el archivo `$audio_title` si los créditos de apertura y el tráiler tienen la misma duración), y modificamos la llamada a Sox de la línea 60 para terminar el trabajo:

```
shell "sox", $audio_title, 2
"audiodump.wav", 2
$audio_trailer, "-o", 2
$audio_total;
```

El silencioso `$video_trailer`, creado desde imágenes JPEG al igual que `$video_title`, se añade al parámetro `$video_file` en el comando `mencoder` de la línea 63. El operador de cámara apreciará ser mencionado en los créditos, y unos enlaces Web pueden dirigir a información más detallada para los espectadores interesados. ■

RECURSOS

[1] Listados de este artículo: <http://www.linux-magazine.es/Magazine/Downloads/63>

SampleRate	0	0
0.000125	0	0
0.00025	0	0
0.000375	0	0
0.0005	0	0
0.000625	0	0
0.00075	0	0
0.000875	0	0
0.001	0	0
0.001125	0	0
0.00125	0	0
0.001375	0	0
0.0015	0	0
0.001625	0	0
0.00175	0	0

Figura 3: La información en bruto de un archivo de audio silencioso de dos segundos de duración.