Fun With Time Lapse

Call me silly but I like to see work being done. There are many times when I've stood back at the end of a large project with visible results and said "man, I wish I had a recording of that". I keep saying that I wanted to setup a webcam to take pictures for a time lapse and today I finally did.

The What:

- Logitech Pro 9000
- Old laptop running Linux
- Tripod

The How:

Hook up the webcam to your laptop and point it where you want to take the shots. I recommend using `camorama` to get things lined up since it will show you the real-time video. If you only want to take a shot every minute you could also use it to take the pictures as it has a timer built-in. However, you can't customize your image type or quality. I chose to use `fswebcam` to capture my images. There are at least a dozen different apps out there that will grab images for you so just choose what works best for you. You can even use FFMPEG to grab images if you'd like.

Once you have your shot framed you'll want to start grabbing images. I wrote the following simple script to grab images:

/usr/local/sbin/camshot.sh

```
#!/bin/bash

JPG_PATH="/home/camuser/Pictures/$(date '+%Y.%m.%d-%H:%M:%S').jpg"

fswebcam \
  -q \
  -p YUYV \
  -r 1600x1200 \
  -S 15 \
  --no-banner \
  $JPG_PATH

chown camuser: $JPG_PATH
```

> You may have to adjust the flags to suit your camera. For example, the `-p YUYV` and `-r 1600x1200` flags will only work on higher-end cameras.

Depending on your setup you may not have to run the image capture as `root`, but I didn't feel like dinking with it and the old laptop I used has nothing on it but these frame grabs.

For my camera the `-S 15` was absolutely critical. That option instructs `fswebcam` to discard the first 15 frames after initializing the webcam before saving a shot. This gave the camera time to adjust the focus and levels on the image. Without this every image I took was washed out.

> Make sure that if you modify the scheme for naming the images that you ensure they can be listed in chronological order easily (such as with a simple `ls`).

Once you have a script that will grab and save an image, schedule it in `cron`:

```
# Take webcam snapshots every 10 seconds-ish
* * * * * root /usr/local/sbin/camshot.sh
* * * * * root sleep 10 && /usr/local/sbin/camshot.sh
* * * * * root sleep 20 && /usr/local/sbin/camshot.sh
* * * * * root sleep 30 && /usr/local/sbin/camshot.sh
* * * * * root sleep 40 && /usr/local/sbin/camshot.sh
* * * * * root sleep 50 && /usr/local/sbin/camshot.sh
```

I chose to take an image every ten seconds. It's really a matter of preference how often you take images. My take on it is that you can always discard images but you can't use what you don't have. So as long as you have the hard drive space, the more the merrier. Each of my captures was around 250k.

After you've gathered your shots you can convert them into a video. This is where it can get confusing because you have so many options. I used FFMPEG (`avconv`), but many other people have had success with `mencoder` and various ImageMagick tools (and other tools on Windows and OS X). If you're going the FFMPEG route you first need to rename your images in a sequential order:

```
$ cd /home/camuser/Pictures
$ mkdir ffmpeg
$ i=1
$ for a in *.jpg; do
>   mv $a ffmpeg/`printf "%04d" $i`.jpg
>   ((i++))
> done
```

This will go through each image and move it to a directory called `ffmpeg` with a zero-padded name (such as `0001.jpg`) in sequential order (assuming you followed the note above regarding your file naming scheme). You can then spit out a movie with a command such as this:

```
$ cd ffmpeg
$ avconv -f image2 -i %04d.jpg -r 24 -s uxga -vcodec libx264
~/Videos/lapse.mp4
```

The command above generates an MPEG4 video using H.264 encoding at 1600x1200 (the same resolution the pictures were taking at) using 24 frames (pictures) for each second. If your picture resolution is different you will want to alter the `-s uxga` flag (you can specify a resolution instead of a name). `Avconv` is supposed to match the resolution of the output to that of the input by default, but that was not working for me. If you want to change the frames per second adjust the `-r 24` flag.

> The position of the arguments is important in `avconv`. For example, if you move the `-r 24` flag so that it preceeds the `-i` flag then it becomes an input parameter instead of an output parameter.

If you want your video to play slower, decrease your `-r` value.

The wife doctored up my resulting video in Adobe Premiere, but here's the result of our Saturday:

`Avconv` is an extremely powerful tool. You can use a command such as this to resize your video:

```
avconv -i ~/Videos/lapse.mp4 -c:v libx264 -q:v 5 -vf scale=640:-1
~/Videos/lapse640.mp4
```

Or if you want to convert it to Flash Video:

```
avconv -i ~/Videos/lapse.mp4 -c:v libx264 -q:v 5 -vf scale=640:-1
~/Videos/lapse.flv
```

Enjoy!