Development Manual

Cliff Swafford, John Hughes, Satyaraj Ryali

February 20, 2011

1 Recorder Program Flow

1. **Build A/V container**

   (a) Using av_guess_format, our program attempts to guess the audio and video codec based off of the file input. With the format received from recorder, we build the format container using avformat_alloc_context. Then, we open the container for use with the recorder.
   Note: We only handle the mkv container.

   (b) Now that we have the codec type required for our containers, we can intialize the audio and video streams. Our audio and video streams configured for **Mpeg4** video encoding and **MP2** audio encoding, respectively.

2. **Camera Initialization**
   Open up file descriptors to the webcam for video and audio recording. Initialize formats for the data that we receive.

   (a) Using OSS, we open a file descripter to **/dev/dsp/** for audio recording. We sample at a rate of `44kHz` with `16` bits per sample. Our audio encoding format is **MP2**.

   (b) video_record_init Open the camera at **/dev/video** for communication. Set the format of the video frames received from the camera to be **YUY2** encoded. Initialize the width and height of the frame to **640** by **480**.

   (c) Since the webcam doesn't enable read() access, we need to mmap the webcam frame buffers into our programs address space. Currently, we are using **10** buffer frames. Our buffer structure is shown as described below.

```
struct buffer {
    void * start;
    int length;
};

struct buffer * buffers;
```

   (d) We then instruct the webcam to queue 10 buffer frames to be ready for retrieval.

(e) SDL Initialize

- In order to display the webcam's image to the screen, we use SDL to display the images we receive from the webcam buffer. We initialize the SDL window to match our image format paramters and create a **YUYOverlay** to present the frame.

3. **Concurrent Encoding**
   In order to prevent process blocking from audio and video encoding, each encoding function is spawned on its own thread. Write access to the container is proteced via mutex locks.

   (a) **Video Recording**

   i. video_frame_copy Dequeue a frame from the webcam buffer into our buffers struct and return the index of the buffer that is available.

   ii. video_frame_display Use the index of the newly available buffer and memcpy the video frame into the SDLYUY Overlay.

   (b) **Video Encoding**

   i. Now that a frame from the webcam is now available in our program, we send it to `video_frame_compress` for encoding.

   ii. In order for `ffmepg` to encode our video frame, we need to convert it from a `yuyv422_frame` to a `yuyv420_frame`.

   iii. Now that our frame is converted, we build a video packet. Then we scale its presentation time stamp with codec specifications and our actual encoding rate.

   iv. After the packet is intialized and filled with our frame data, it's sent to be written to the file.
   Note: This write operation is protected by a mutex to prevent concurrent writes.

   (c) **Audio Recording**
   audio_segment_copy Read one sample from the microphone.

   (d) **Audio Encoding**

   i. audio_segment_compress Using the sample read from the microphone, we send the data to be encoded into an audio packet.

   ii. audio_segment_write Our audio packet is then written to the container.
   Note: This write operation is protected by a mutex to prevent concurrent writes.