

# Assignment 1

## CS414, Multimedia Systems (Instructor: Klara Nahrstedt)

Posted: January 28, 2009

This is the first assignment for your Peer-to-Peer Video-on-Demand( P2P-VOD) project where you experiment with the access to audio/video devices with your group members. This assignment will have three goals: (a) find audio/video content on the web, and (b) build a player for the audio and video files. All assignments will be implemented on the Linux Dell Machines in 0216 SC lab.

### Assignment Description:

#### 1. Find Audio and Video Content

The goal will be to find your content for your P2P-VOD project on the web. You should find up to 10 audio and video files that you will consider storing in your video server. The audio files should be of different content ranging from music to spoken word of a newscaster, and the video files should be of different content ranging from dancing and action scenes to newscasters talking heads. The audio and video files should be no longer than 3 minutes. The found content on the web should be of specific formats including Motion JPEG videos and .au (or .wav) audio format to make it easier on you in later assignments for streaming protocols since your player must have a detailed control over the frames and samples in your video and audio files.

#### 2. Playback of the audio and video files

Implement a program *p2p-vod* which takes arguments from a Linux command line. The main arguments for *p2p-vod* functions such as **PLAY** and are "**function, type, duration, file**". For functions such as **REWIND** and **FAST FORWARD** there will be additional parameters such as "**step**". The compression format for video should be Motion JPEG (MJPEG).

1. "**function**" argument will have values **PLAY**, or **REWIND**, or **FAST FORWARD**,
2. "**type**" argument will have values **audio**, or **video**,
3. "**duration**" argument will specify the duration in **seconds** how long you want to play/rewind/fast forward your clip,
4. "**file**" will be either **file.au** (audio file), or **file.mjpg** (video Motion JPEG).
5. "**step**" parameter for the functions **REWIND** and **FAST FORWARD** means the number of data units you want to skip when doing rewind or fast forward functions, i.e., you will read every 'k' video frame backward or forward within the video file.

Examples below show the call of the individual arguments.

- The command line called: **p2p-vod PLAY audio 5 meeting.au** should retrieve from a local file **meeting.au** the **audio** information and write (**PLAY**) it to the audio output device (headphones !!!!, not speaker) for **5 seconds**.

- The command line called: **p2p-vod PLAY video 5 football.mjpg** should read Motion JPEG compressed video from the local file **football.mjpg** and write the video frames to the screen. The duration of the operation is **5 seconds**.
- The command line called: **p2p-vod REWIND video 5 football.mjpg 3** will open the video file **football.mjpg**, position the read pointer at the end of the file, and start to read and display **every 3th** (step is 3) frame backwards towards the beginning of the file. After **5 seconds** the rewind process should stop. It means that the reading process skips 2 frames and reads the 3th frames, skips another 2 frames and reads the next frame, etc. (e.g., if the file has 20 frames, then the rewind process reads and displays the following frames in the specified order: 20, 17, 14, 11, 8, 5, 2).
- The command called: **p2p-vod FF video 5 shoe-store.mjpg 6** will open the video file **shoe-store.mjpg**, position the read pointer at the beginning of the file, and start to read and display **every 6th** (step) frame forward (towards the end of the file) for **5 seconds**.

## Comments

- The **REWIND** and **FAST FORWARD (FF)** functions should be implemented *only for video*. This is consistent with the real VCR applications where when rewinding or forwarding a movie, the audio is left out, you see only some video frames on the screen.
- It is strongly recommended that you parse through the video file and find where each frame starts (or you invoke a function that allows you to create *index file* of your video file). Then you create an *index file* for this video file which will include the sequence of pairs of information (frame number, starting file position of the frame in bytes). For example: index file of a video file which has 5 frames would be : (1,1), (2, 4500), (3, 9005), (4, 13000), (5, 17300) (Note: Keep in mind that you work with Motion JPEG compressed video frames, hence each video frame might have a different size, although the size will not very much differ in this compression scheme.) This index file will then help you to implement the rewind and fast forward functionality, because as you skip every 'step' frame, in the index file you can find immediately the position of your frame that you need to display and you can read the frame immediately instead of searching for the beginning of each frame during the rewind/fast forward operations.
- The following *websites* contain source code libraries and/or examples that might be useful for *Linux audio*. Although not strictly required, you may choose to download and install one or more of these packages for use in the audio component of your program. You are also free to use source code from other sources as long as you understand it thoroughly and document it well with proper attributions.
  - Simple DirectMedia Layer < <http://www.libsdl.org> >
    - Initialization < <http://www.libsdl.org/intro.en/usingsound.html> >
    - Audio < <http://www.libsdl.org/intro.en/usingsound.html> >
  - Sound eXchange < <http://sox.sourceforge.net/> >
  - Advanced Linux Sound Architecture <[http://www.alsa-project.org/main/index.php/Main\\_Page](http://www.alsa-project.org/main/index.php/Main_Page) >
  - Java Sound Library <<http://www.jsresources.org/links.html>>
- The following *websites* contain source code libraries and/or examples that might be useful for *Linux images/video*. Although not strictly required, you may choose to download and install one or more of these packages for use in the video component of your program. You are also free to use source code from other sources as long as you understand it thoroughly and document it well with proper attributions.

- FFmpeg < <http://www.ffmpeg.org/> >
  - Please check out the hints at <<http://www.cs.uiuc.edu/class/cs414/help/cs414Hints.htm>> for more information
  - Useful ffmpeg tutorial <<http://www.dranger.com/ffmpeg/>>
  - Useful ffmpeg manipulation < <http://howto-pages.org/ffmpeg/>>
- Java Media Framework
  - <<http://java.sun.com/javase/technologies/desktop/media/jmf/>>
  - **Please note that this software may not be available in CSIL-0216**
  - Programmer Guide < <http://java.sun.com/javase/technologies/desktop/media/jmf/1.0/guide/index.html>>
  - JMF Samples and Apps <<http://java.sun.com/javase/technologies/desktop/media/jmf/2.1.1/solutions/>>
- GTK+ <<http://www.gtk.org/>>
- If you plan to use C/C++ for the MPs, ffmpeg and SDL are recommended. Please go through all the ffmpeg tutorials, read the documentation and inspect the code. The tutorial might seem to cover most of this MP but please start early. Also, please make sure you understand each piece of code in the tutorial, as they will be used to develop later MPs.
- If you plan to use Java, Java Media Framework might be an option. Please check out the JMF programmer guide and sample codes. For this MP, JMF may be overkill as most of the functions might be implemented in those tutorials and examples. However, please keep in mind that future MPs require to have individual frame access/manipulation. Thus, please make sure you know how to do that in JMF (e.g. implementing a new media protocol, a new data source or a new codec/decoder). **It is emphasized that JMF has not been installed yet in the csil-0216 machines due to some platform incompatibility.** We are still working with TSG on that. If you really want to use JMF, please let us know.
- Another option for Java is to build from scratch without using JMF. Essentially, MJPEG is just a sequence of JPEG images. Each frame is essentially a JPEG image with associated audio track. JPEG and AU format are supported by standard Java sound/graphics libraries (e.g. Swing, Java Sound Library). Please keep in mind that even though this approach fully allows individual frame access, synchronization might be an issue. Thus, please make sure you have a rough idea on how to do audio/video synchronization.
- Please start early by reading the documentation and inspecting the example code. Also, please follow the project hints, which might include sample audio/video files for testing and example source code at a later date, at <http://www.cs.uiuc.edu/class/cs414/help/cs414Hints.htm>.

## Delivery

Each group delivers:

- source C, or C++, or Java of your **p2p-vod** program in the particular group directory. The source code evaluation will be based on how well is your code documented. If you use some code you found on the web (You must understand the code you found and include in your code, not just blindly copy the code !!!) or in a local system directories, document it. It is very important that

you give credit to people who developed the previous code. Your own code should include the following information at the beginning of each C/C++ /Java file.

- Each major source file should include
  - File Name: Name of the File
  - Description: Short description what the file includes (general description, what kind of functions are embedded in the file).
  - Version: version of your code. You start with version 0 and as you improve the code, at some point you increase the version.
  - Programmer's Name: your name(s) who developed the code
  - Company/University Name: you put the name of the course, department and university you implemented the code for;
  - Date:
  
- Each function in your C/C++ /Java file should have a header with information:
  - Function Name: Name of the Function
  - Description: Short description what the function does.
  - Arguments: Specification of each input argument parameter entering the function and its description.
  - Results: Specification of returning parameters exiting the function and their description.
  - Comments: some special system issues connected with this function
  
- Group representative(s) comes at the scheduled time (we will have a sign-up sheet) between **5pm and 7pm on Monday, February 9th** and shows a demo of the required programs in 0216 Siebel Center.

## Evaluation of the Assignment (100 Points)

- **Find 10 Audio and 10 Video Files – 10 points**
- **PLAY AUDIO**- demonstration **10 points**, correct answer to questions for this part **5 points**;
- **PLAY VIDEO** - demonstration **20 points**, correct answer to questions for this part **5 point**;
- **REWIND VIDEO** - demonstrations **15 points**, correct answer to questions for this part **5 points**;
- **FAST FORWARD VIDEO** - demonstration **15 points**, correct answer to questions for this part **5 points**,
- **DOCUMENTATION - 10 Points** - Each group should write a short README file (1-2 page) in pdf format. In this documentation file you should specify your design/implementation document, short description of each functionality in terms of its implementation - don't repeat what is in this assignment writeup). Email this document to the TA [hnguyen5@uiuc.edu](mailto:hnguyen5@uiuc.edu), and also store it in your group directory.

The demonstration of the whole assignment for one group should take no more than 20 minutes.