Worksheet – Audio Processing > Wave Files

Goals:

To create an expandable music player that will be able to play pcm wave files.

Introduction:

C++ has a useful organization tool called "classes" built into it. These allow you to create new types with their own subsets of functions and properties easily. This assignment requires you to use this tool to create a music player that initially plays wave files, but can be easily expanded. To this effect, you will need to create three files: main.cpp, which will be used as a test file, sound.cpp to hold class function implementations, and sound.h to describe your class.

Before you jump in, notice the library files indicated on the last page, they may be useful for this assignment. You may or may not have used the /dev/dsp/ before, if not review the following site page, or for a summary check at the end of this assignment: http://4front-tech.com/pguide/audio.html

Now the meat of the problem, if you dump a wave file directly into the dsp you will receive noise and possible ear damage as a just reward. The good news is the actual data is in a format that dsp can handle, the bad news (more bad news) is that you need to parse the header, feed the header information into the dsp, and then you can feed in the data chunk. Use the following website for information on wave file header format: http://www.sonicspot.com/guide/wavefiles.html

An optional task, but a fairly easy one, is to define the header of a new file based on an input file and record a from the dsp.

Tasks:

- Create a sound.h file to describe a class that contains the header information for a wave file and functions to parse a header and play a loaded wave file.
- Create a sound.cpp file to implement the sound class
- Create a main.cpp file to test your class
- Optionally, record from the dsp.

Additional Information:

DSP info:

You must manually set the following:

```
  // The following code (block is the block aligned value) sets the dsp's format
  int format;
 if(block == 2){
 format = AFMT_S16_LE;
 } else if(block == 1){
  format = AFMT_U8;
 }
 if (ioctl(fd, SNDCTL_DSP_SETFMT, &format)==-1){
  perror("SNDCTL_DSP_SETFMT");
  exit(1);
 }

 // The following code sets the number of channels
 if (ioctl(fd, SNDCTL_DSP_STEREO, &channels)==-1){
  perror("SNDCTL_DSP_STEREO");
  exit(1);
 }

 // The following sets the sampling rate
 if (ioctl(fd, SNDCTL_DSP_SPEED, &rate)==-1){
  perror("SNDCTL_DSP_SPEED");
  exit(1);
 }
```

Useful Libraries:

```
#include <stdio.h>
#include <stdlib.h>
#include <strings.h>
#include <string.h>
#include <math.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/wait.h>
#include <sys/ioctl.h>
#include <fcntl.h>
#include <linux/soundcard.h>
#include <sys/ipc.h>
#include <sys/shm.h>
#include <iostream>
```