Problem Set 7 – Playout of WAV Files

Total points: 60
Create a program that plays a selected one of several WAV files to the laptop speaker. Program uses:

- sndfile library to read WAV files
- PortAudio library to play audio data to speaker using a callback function
- ncurses library to read key presses without waiting for CR

Use the structure in file play_wavfiles.c and fill in code under the comment blocks Problem 1, 2 and 3.

libsndfile, libportaudio
You should have these two installed. If not, see ps5 and ps6 instructions.

Ncurses
On MAC
  ncurses is already installed
On PC
  Start cygwin_setup and search for curses. Install

Wav file reference
http://soundfile.sapp.org/doc/WaveFormat/

libportaudio reference
http://www.portaudio.com/

libsndfile reference
http://www.mega-nerd.com/libsndfile/

C function reference
The following URL provides a good reference for C language library function usage:
http://www.cplusplus.com/reference/cstdlib/

Makefile tutorial
http://www.cs.colby.edu/maxwell/courses/tutorials/maketutor/

Problem 1: Parse command line and open all files (20 points)

Create a program that has the following command line usage:

```
    main ifile_list.txt
Where
    ifile_list.txt is an plain text file that contains a
    list of WAV audio files that can be played
```

All files must have the same sampling rate and same number of channels and be no more than MAX_CHN channels.

Parse command line and open input file
Parse the command line. If parsing fails, print an error diagnostic and exit. If successful, open ifile_list.txt and read each line which is a path to one of the WAV files that can be selected and played. Print list of input files paths with a number indicating order in the file.

```
    Input files:
        0    file1.wav
        1    file2.wav
```

(etc.)

Open WAV files
Use the libsndfile library to open WAV audio files and read the SNDFILE header of each input file. Use error checking and error reporting in all operations.
Members of the SF_INFO structure that you will want to access are:

```
    sfinfo.samplerate;
    sfinfo.channels;
    sfinfo.format;
```

Check that all files have the same sampling rate and same number of channels. If not, print an error and exit.

Problem 2: Initialize structure and allocate buffers (10 points)

Initialize data structure
The example code shows a structure Buf:

```
    struct BUF_tag {
        /* libsndfile data structures */
        SNDFILE *sndfile[MAX_IFILES];
        SF_INFO sfinfo[MAX_IFILES];
        unsigned int num_chan;
        int selection;
        float *buffer;
    };
    typedef struct BUF_tag Buf;
```

The main program declares iBuf of type Buf and a pointer p to iBuf. Initialize the value `num_chan`. The variable `selection` indicates to the callback which file should be played out. Initilize this to -1 which causes callback to load zeros into the D/A output buffer.

Use ncurses example code to receive key presses and take action. Possible actions based on key pressed are:

1 … N Switch to playing input file corresponding to the number pressed.

Q Quit the program

If Quit, then close all files and free all allocated storage before exiting.

In the code, `iBuf.selection` functions as a one-way information link (a 1-element FIFO) between the main thread and the callback thread. This eliminates the possibility that read/write race conditions might cause undesired output.

Problem 3: Play file to audio output (30 points)

Use the PortAudio library callback function to enable playing the buffers of input file to D/A.

In Callback

Fill output buffer from file data associated with selected WAV file.

In the body of the callback,

- Read the selection variable
- If selection is -1, fill output buffer with zeros.
- Otherwise, selection is the index of the sndfile data structures and hence selects a file to play. Use `sf_read_float()` to read `framesPerBuffer*num_chan` float-valued audio samples from the read position of the selected file into the callback output buffer.
- If the returned count from `sf_read_float()` is less than `framesPerBuffer*num_chn`, execute `sf_seek(sndfile, 0, SF_SEEK_SET)` to rewind to the start of audio data and then do another `sf_read_float()` to read sufficient floats to fill the reminder of the buffer.