Lab 1 - Create a DSP Frontend Application

February 19, 2022

1 Overview

In this lab, we will create a DSP frontend application. This prepares us for subsequent labs which involve extending the code to perform signal processing on the input to the frontend.

2 Lab Exercise

2.1 Parse Input and Output Filenames

The aim of this exercise is to write a C frontend that takes in an input filename and output filename and then prints them out on the terminal.

An example of how the frontend should behave is shown below:

```
> dsp_frontend -i input.wav -o output.wav
Input file is input.wav
Output file is output.wav
```

As suggested above, the program should be named `dsp_frontend`. To get started, created a file called `dsp_frontend.c` and insert the following code.

```c
#include <stdio.h>

int main(int argc, char* argv[])
{
    /* Parse command line (skip executable name) */
    argc--;
    argv++;

    if (argc == 0)
    {
        /* Print usage information */
        printf("Usage: dsp_frontend -i <input_filename> -o <output_filename>\n");
    }

    /* TO DO: Insert filename parsing code here */

    /* TO DO: Print input and output filenames here */
```

```
    return 0;
}
```

Some pointers for completing the exercise include:

- Getting familiar with the contents and usage of `argc` and `argv[]` of the `main()` function
- Understanding `if` statements and `for`/`while`loops

2.1.1 Building dsp_frontend

To build `dsp_frontend`:

- **On Windows**, create a C++ project on Visual Studios and add in the `dsp_frontend.c` file. Once built, the `dsp_frontend` executable can be found in your project folder.
- **For OSX**, refer to the `Makefile` example in Lecture 1 to build the program. Alternatively, create an Xcode project and add in the `dsp_frontend.c` file.

2.1.2 Testing dsp_frontend

To confirm that a program is working as desired, testing is always required. Have a think about the different test cases that can be used and write brief notes on the reasoning behind each.

2.2 Make Output Filename Argument Optional

Once the program is behaving correctly for the previous exercise, implement extra logic to prescribe a default output filename whenever `-o` is not detected in the arguments provided. The following example demonstrates the new behaviour:

```
> dsp_frontend -i input.wav
Input file is input.wav
Output file is output.wav
```

```
> dsp_frontend -i input.wav -o custom_output.wav
Input file is input.wav
Output file is custom_output.wav
```

Note how the original behaviour should not change and the default output filename should be set to `output.wav`.

To illustrate the added functionality, the previous function usage description may be replaced with the following:

```
    #define OUTPUT_FILENAME_DEFAULT ("output.wav")

    ...

    printf("Usage: dsp_frontend -i <input_filename> [options]\n");
    printf("options\n");
    printf(" -o <output_filename>\n");
```

```
printf("         Output filename.  If not specified defaults to %s\n"
      ,OUTPUT_FILENAME_DEFAULT);
```