**ECE 3822: Engineering Computation II**

# Homework No. 7: File I/O

**Goal:** The goal of this homework is to demonstrate how to do various types of file I/O in C. You must use fopen, fclose, fread, fwrite, fprintf, and fscanf for these problems. No other functions or methods are acceptable. Deviation from this requirement will result in a failing grade.

**Description:** The tasks are:

1. Create a file called files.list that contains all the \*.txt files in the EEG report database we have been using in class. Split this into three equal-sized files: file1.list, file2.list and file3.list. (If you cat these files together, they should match the original file.)
2. Write a C program with the following interface:

my\_cat file1.list –help file1.list file2.list file3.list file1.list ...

If the “-help” option is placed anywhere on the command line, the program displays a help message. If any of the arguments are wrong (e.g., no files are specified), a help message is also displayed. In other words, you should catch all combinations of incorrect usage of the program and display the help message.

The program loops over all file lists and opens each file, reads the file line by line, and prints the corresponding filename to stdout, and then displays the output of each file. Your output could look like this:

Opening file1.list:

000000: firstfile.txt

\*\*\*\*\*\*

[the contents of firstfile.txt are displayed here...]

\*\*\*\*\*\*

000001: secondfile.txt

\*\*\*\*\*\*

[the contents of secondfile.txt are displayed here...]

\*\*\*\*\*\*

...

Be sure to close all file pointers that you open so the program does not run out of memory. Also be sure strip off any linefeed characters that you read at the end of each line. Demonstrate that your program can run through the entire EEG report database by using head, tail and the time command.

1. Write a C program called my\_write that does the following:
* Create a binary file (open in write mode)
* Write 100 signed short integers ranging from -25000 to +25000 to the file as two-byte values (NOT ASCII values).
* Close the file

Similarly, write a C program called my\_read that reads the values written to this file and displays them using fprintf. Demonstrate that the data you are reading with your read program matches what you wrote and matches the output of od **EXACTLY**. Use ls to show that the file size is correct.

1. Modify your program in step 3 to write a signal one second in duration that consists of the sum of three sinewaves (frequencies = 500 Hz, 1000 Hz, 2000 Hz; amplitude = 5000; phase = 0) sampled at 10,000 Hz. There should be exactly 10,000 samples in this file (20,000 bytes). Using your read program, show that you get exactly the same values that od displays. You might also check your result using MATLAB. Argue that the values you are seeing make sense based on what you learned in ECE 3512.
2. Write two programs – one that writes binary data and one that reads binary data. The first program writes the following data to a file:

char a = 27;

short int b = 32767;

unsigned short int c = 32769;

float d = -27.2727272727272727272727272727272727272727272727;

double e = -27.2727272727272727272727272727272727272727272727;

short int f[10] = {0, 1, 2, 3, 4, 5, 6, 7, 8, 9};

float g[10] = {0, 1, 2, 3, 4, 5, 6, 7, 8, 9};

double h[10] = {0, 1, 2, 3, 4, 5, 6, 7, 8, 9};

The second program reads this data and displays it to the terminal using fprintf.

Demonstrate that your write and read programs work correctly. Be sure to show that the file size is correct. Use od to show that you are getting the correct result. **Explain any discrepancies that you observe** (e.g., are the floating point values represented exactly when you read them back from the file?). Based on your observations, explain why it makes sense to write audio data as 16-bit signed integers rather than ASCII of floating point numbers.

Note that if you write the data as ASCII, there is a chance your read and write programs will work, but the file size will not be correct. There, you must demonstrate you are reading and writing the data as binary values.