CS 330 Homework

Trois Voix: Managing Memory

1 Overview

Your responsibility in this homework is to generate a WAV file containing the English pronunciation of a given integral number.

2 Requirements

In order to complete this homework, please satisfy the following requirements:

1. Write a program that, given an integral number, generates an audio file containing the pronunciation of the number according to standard English rules. See the `trois_voix` directory on the `W` drive for examples.

2. Do not generate the audio directly. All the audio pronunciations for individual number words like *five*, *trillion*, and *negative* are provided to you on the class `W` directory. Read one in as necessary from the W drive, extract the audio samples, concatenate them with what you've built up so far, and write out a new WAV file. (Reference the provided WAVs only; do not read copies of these files from other locations, as this will break your code when we go to grade it.)

3. Assume an integral number n is passed as the first command-line argument and the output WAV file as the second, e.g., `./trois_voix -123 one_hundred_twenty_three.wav`. Further assume that $-999,999,999,999 <= n <= 999,999,999,999$.

4. Create all source code and output files in a directory named `trois_voix`. Write all C functions but `main` in file `trois_voix.c`. A corresponding header file is provided to you on `W`. Put your `main` function in `main.c`. Write a makefile that compiles `trois_voix.c` and `main.c` to an executable named `trois_voix`. Compile with flags `CFLAGS = -std=c99 -Wall -g`.

5. Implement the functions declared in `trois_voix.h` on the W drive. C has no mercy on random code generators like us. So that we may test individual pieces of your solution, conform to the documented interface.h= on the `W` drive. The functions declared in this file will be individually tested.

6. Your program must compile without warnings. When run under

   ```
   valgrind --leak-check=yes --show-reachable=yes
   ```

   your program must report no abuses of memory, e.g., memory leaks, accesses to uninitialized memory.

7. Zip up your directory to `trois_voix.zip` with the following:

   ```
   (cd .. && zip -r trois_voix.zip trois_voix)
   ```

   This command assumes your current working directory is `trois\_voix`. The *.zip file is placed in the parent directory.

8. To test your code, run the following:

   ```
   (cd .. && ~/w330/trois_voix/test_trois_voix)
   ```

   This script only tests a few things like proper names and basic functionality. This script does not test all requirements. You need to do your own testing too. Failure to do so will likely result in a rejected submission. After you pass these tests, the script will prompt you to submit and notify us of your submission.

3 Meta

The reference implementation of this assignment consumed about 300 lines of code, not counting documentation. Standard library functions used included `strcpy`, `strcat`, `free`, `strchr`, `strncpy`, `malloc`, `memcpy`, `fopen`, `fprintf`, `fread`, `fwrite`, `fclose`, and `sprintf`.