Design Plan - Hangman

James Viner

Project Summary

The task was to recreate the word-guessing game of hangman in the console, reading in a custom word bank from a file in the user's home directory and saving end-of-game statistics to a different file. File input/output and user-interaction were the biggest portions of this project.

Challenges

I will be upfront: this project was significantly more complex than it originally presented itself as. My pre-project design document didn't end up containing half of the complexity of the final product as a result of me underestimating the amount of error checking and edge-casing involved with working with input files that may or not be sanitized and input from the user in a similar vein. The initial act of just reading words from a file and validating them took me more than an afternoon alone to get working, and it was hacky and barely functional.

Working with dynamic memory proved to be an interesting challenge, though it mainly just ended up bogging down my code with free() statements that disrupt normal flow. It would've helped to have made a function that closes any open files and de-allocates memory dynamically based on needs, but I didn't end up having the time or energy to figure that one out.

Time management was rough this time around. I spent the entirety of Thursday afternoon until 2 AM working on the project, slept until 7, and then I worked from 8 until the time of writing this post-project write-up 22 hours later. I have some experience before this course with programming in C, but not enough familiarity to take the jump in complexity from mastermind to hangman in stride. If this project was intended to humble me, it did its job.

Successes

The interesting thing about this project was how well certain portions went in comparison to others. The actual designing of the hangman game loop itself was a lot easier after having worked on mastermind, and I learned my lesson about handling Ctrl + D signals from the user. Where I spent a day and a half working on file input and output, I only spent a couple hours working on the game's logic and mechanics. Admittedly, some of that came down to what I believe is a very smooth implementation of the hangman

letter masking mechanic by using an array of booleans to correspond to which letters should be turned on or off, but the point stands.

Certain portions of code during the implementation of additional features went through significant rewrites to reduce their complexity and to increase their efficiency, and I believe I'm getting better at refactoring the sort of hacky "just for now" solutions that I come up with at first glance into more efficient and self-contained functions.

Lessons Learned

Pointers are, surprisingly, not actually that hard to figure out in most cases, and when in doubt: the compiler usually knows best. On that note, using pointers to return values from functions and then de-allocating them later with free() is actually a pretty simple way to get proper returns without having to pass and modify arguments.

Taking regular breaks to avoid burn-out during longer sessions of coding is essential. I would come back after 15-20 minutes of just pacing or taking my mind off of the project refreshed and with a fresh perspective on many of the problems I approached.

Finally, file error handling combined with dynamic memory allocation makes for a genuinely miserable experience. I have to say this was a massive increase in difficulty as compared to the last project, although now that I am at the other end of it with a (hopefully) working and resilient program, I do have to say that it feels good to have been challenged. Honestly though, I am finishing this line on my 23rd hour without sleep and the moment I commit this document I am falling asleep. I am glad to have done the project, but I am even more glad to be done with it.