CS214 Project 0

Basic Data Sorter

Brian Huang & Matthew Stone

Design

Our program is designed to receive comma-separated values of imdb movie records through stdin, a "-c" argument to signify sorting by column, and a parameter defining the column to sort. An example command would be:

```
cat movie_metadata.csv | ./sorter -c movie_title
```

Keep in mind that the *sorter.c* **file has to be made into an executable for the above command to work by using a command such as:**

```
gcc -o sorter sorter.c
```

The *sorter.c* **file contains the majority of the function of the program. This file takes in the movie records and stores it as an array of structs, calls the mergesort function, and outputs the sorted array of structs.**

In addition to the sorter.c file, our project contains *mergesort.c* **and** *sorter.h***. The** *mergesort.c* **contains the necessary functions to sort movies according to the user defined value. The** *sorter.h* **is a header file that contains a struct array definition detailing how our program stores movie records. This header file also prototyped functions, such as mergesort, to be used in our program.**

Assumptions

Initially, we underestimated the difficulty of this project. The concept of the program seemed straight forward, however neither of us had experience in C. We both have most of our coding experience in Java, so some tasks in C proved to be more complicated and time consuming.

Since we learned and used mergesort in past Java classes, we found it fairly easy to implement mergesort into our sorting program.

Difficulties

Due to no prior experience in C, our greatest difficulty was managing memory and referencing points. We overcame this mainly through trial and error as well as using our knowledge for the classes lectures.

One large issue we encountered was using our tokenizer correctly when tasked with titles containing commas. To correct this we added additional tokenizers to to split up a line when there was a title containing commas.

Another major issue was getting segmentation faults due to our inexperience with memory allocation. We overcame this by learning how to use a designated size_t variable to keep track of how big the malloc was and using said variable to realloc when the initial malloc was not enough memory.

Testing Procedure

We hosted our project on github and would test our project before pushing it to master branch. We used a small csv file with a few lines for basic testing in the early development of the project, however most of our testing was done using the provided csv file.