CS214 Project 1

A Multi-process Sorter

Brian Huang & Matthew Stone

Design

Our program is designed to sort all properly formatted csv files in a given directory or current working directory and output a new csv files for each input to a given directory or the source files directory. There are three possible arguments to give when the program is called: the column to sort by, the input directory, and the output directory. Directories can be entered as either a relative or an absolute directory. The column to sort by is referenced by a -c argument, the input directory by a -d argument, and the output directory by a -o argument. The only argument that is required is the column to sort by. The column to sort by argument has to before the input directory and the input directory has to be before the output directory. Some different ways to run the program are as follows:

Sortings files in the current working directory by the movie_title column (this is the one without directory arguments and outputs the sorted files in their respective directories:

```
./sorter -c movie_title
```

The full command for this project would be:

```
./sorter -c [column header] -d [input directory to sort] -o [output directory]
```

Some possible commands:

```
./sorter -c movie_title -o [output directory]

./sorter -c director_name -d [input directory]

./sorter -c budget -d [input directory] -o [output directory]

etc...
```

The *sorter.c* file contains the majority of the function of the program. This file creates the processes for files and directories and sorts the csv files.

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

We knew this was going to be hard because this was the first time we dealt with multi process programs and the fork function. It took us several attempts to implement fork(), wait(), WEXITSTATUS() functions correctly. Our sorter was successful on our last project so we assumed that there would be very little change needed on the basic sort functionality. We only had to edit a few lines pertaining to sorting to change from printing to stdout to the processes individual new file. The entire mergesort.c file went unchanged.

Difficulties

This project was very difficult because the new concept of fork, child processes, and the wait function was hard to understand. Another difficulty was getting the count of PID's from nested directories, which we overcame by exiting (exit()) using 1 as the exit argument (exit(1)) and using the WEXITSTATUS(status) and adding it to our child counter. Another difficulty was implement both relative and absolute paths for the directory parameters. There were many other difficulties that we cannot think of off the top of our heads right now.

Testing Procedure

We hosted our project on github and would test our project before pushing it to master branch. We tried many cases such as no arguments, only an input directory argument, only an outputdirectory argument, both input and output arguments, and for all of these cases, we would try both absolute and relative file paths.