CS214 Project 1: Multiprocess CSV Sorter

Abstract
This is the second in a series of projects that will involve sorting a large amount of data. In this second phase, you will write a multi-process C program to sort a list of records of movies from imdb alphabetically by the data in a given column. You will make use of the concepts learned in lecture including file/directory access, forking processes, etc.

Introduction
The basic parts and dataflow for the project are the same as the previous. The CSV file with movie metadata will remain the same. The sorting algorithm will also remain the same. If you properly modularized your code in Project 0, you should be able to reuse almost all of your code.

In this project, you will read in a directory name and walk through the directory structure to find all .csv files. There may be multiple levels of directories that you will need to recurse through. For each directory you find you should fork() a child process that will sort each of the files in it and output the results to a different file. You should NOT use exec for this project. You should write one program that, when copied from the parent to the child process, will continue running. You can use the return value of fork() in a conditional to choose different blocks of code to run within your code. You will want to make sure to prevent zombie and orphan child processes. You will also want to make sure you to not create fork bombs that will bring down machines. In all cases of bad input, you should fail gracefully (e.g. no segfaults).

You will output metadata about your processes to STDOUT. This metadata will show the total number of processes created and the pids of all created processes.

Methodology
a. Parameters
Your code will read in a set of parameters via the command line. Records will be stored in .csv files in the provided directory. As mentioned above, directory structures may have multiple levels and you must find all .csv files. Your code should ignore non .csv files and .csv files that do not have the format of the movie_metadata csv (e.g. comma-separated heading list followed by same numer of comma-separated values in records).

Your code must take in a command-line parameter to determine which value type (column) to sort on. If that parameter is not present your code should output an error, deallocate any memory in use and shut down gracefully. The first flag your program must recognize will be '-c' to indicate sorting by column and the second will be the column name:

./sorter -c food

Be sure to check that the flag exists. Your code should output an error and shut down if the '-c' flag is not present. Be sure to check the column heading specified is in each CSV. If not, output an error, but continue scanning and sorting files.

For this phase you'll extend your flags from one to three. The second flag your program must recognize will be ‘-d’ indicating the directory the program should search for .csv files. If this parameter is not specified, your code should continue without error and search the current directory.

./sorter -c food -d thisdir/thatdir

The third parameter to your program will be ‘-o’ indicating the output directory for the sorted versions of the input file. If this parameter is not specified, your code should continue without error and send its output to the same directoty as the source file.

./sorter -c movie_title -d thisdir -o thatdir

b. Operation

Your code will be reading in and traversing the entire directory. In order to run your code to test it, you will need to open each CSV and read it for processing:

./sorter -c movie_title -d thisdir -o thatdir

Your code's output will be a series of new CSV files outputted to the file whose name is the name of the CSV file sorted, with "-sorted-<fieldname>" postpended to the name.

e.g: 'movie_metadata.csv' sorted on the field 'movie_title' will result in a file named "movie_metadata-sorted-movie_title.csv".

On each new file in a directory you encounter, you should fork() a child process to do the actual sorting.

On each new directory you encounter, you should fork() a child process to process the directory.

To STDOUT, output the following metadata in the shown order:

Initial PID: XXXXX
PIDS of all child processes: AAA,BBB,CCC,DDD,EEE,FFF, etc
Total number of processes: ZZZZZ

You may assume the total number of files and directories will not exceed 255.

c. Structure

Your code should use Mergesort to do the actual sorting of records. It is a powerful algorithm with an excellent average case. You should write your own Mergesort code.

Results
Document your design, assumptions, difficulties you had and testing procedure. Include any test CSV files you used in your documentation. Be sure to also include a short description of how to use your code. Look at the man pages for suggestions on format and content. Do not neglect your header file. Be sure to describe the contents of it and why you needed them. Submit this as “Asst1Doc.txt” or “Asst1Doc.pdf”. Do not submit any proprietary or Microsoft documents.

Submit your “Asst1Doc.*”, "scannerCSVsorter.c", "scannerCSVsorter.h" and "mergesort.c" as well as a “Makefile” and any other source files your header file references as a single “Asst1.tar.gz” file.

FAQ
Q: When sorting by the item which is type of string, the sort rule is A-Z or lexicographic order?
A: The default should be lexicographical order (i.e. how strcmp evaluates strings)

Q: For the column of color, there is a blank before “Black and White”, should it be considered during sorting?
A: For all fields, one should trim leading and trailing whitespace

Q: What about the items with null value? should them be considered during sorting?
A: It is sufficient to move all null items to the beginning of the sorted list, in the order they appear in the original list

Q: Will the grader use a different CSV?
A: The grader will use a different CSV, but it will adhere to the format given in the assignment

Q: Should the sorting be ascending or descending?
A: Ascending (you can always parse a command line parameter to do the opposite)

Q: What are the types of the fields?
A: See listing below.

color
string

director_name
string

num_critic_for_reviews
numeric

duration
dateTime

director_facebook_likes
numeric

actor_3_facebook_likes
numeric

actor_2_name
string

actor_1_facebook_likes
numeric

gross
numeric

genres
string

actor_1_name
string

movie_title
string

num_voted_users
numeric

cast_total_facebook_likes
numeric

actor_3_name
string

facenumber_in_poster
numeric

plot_keywords
string

movie_imdb_link
string

num_user_for_reviews
numeric

language
string

country
string

content_rating
string

budget
numeric

title_year
numeric

actor_2_facebook_likes
numeric

imdb_score
numeric

aspect_ratio
numeric

movie_facebook_likes
numeric