Team EJDC

Eric Jiang

Daniel Chen

Assignment1 - A Multi-process Sorter: Part 1

**Introduction:**

The purpose of this assignment is to obtain a good understanding of how forking works for parallel multi-processing. For the assignment, a directory would be scanned through to find all subdirectories and .csv files that is contains a desired header name (inputted by user). The every level of subdirectories should be scanned through as well (recursion was used for scanning). The subdirectories and csvs should be scanned/sorted in parallel through forking a child process when each is found. The sorted csvs should be placed in given output directory (inputed after optional -o). The current PID should be printed out along with all the children PIDs used for forking and the total # of processes.

**Our design/planning:**

1) Create recursive function search through current directory to find all CSV files/subdirectories.

2) Spawn sorter.c from assignment 0, fork children of sorter into subdirectories. Fork

3) Print the PID, PPID, and counter of total processes.

4) In sorter, output the new sorted CSV files into output directory or current directory.

5) Merge the sorter and the recursive function to work correctly.

**Assumptions:**

* If output directory is not given, then output the sorted file into the current directory.
* Hidden files can be disregarded

**Difficulties:**

Forking was difficult to implement properly. After a recursive function was created for forking, wait(NULL); was used after scan loop to fork. However, this caused a hanging issue as the children would still not terminate due to recursion. Instead a wait loop was made until the childrens all equaled to terminating NULL. Another issue was creating a counter and retrieving the PIDs inside the recursive forking. Since forking itself creates child processes that would disrupt the counter, a separate counter function (that didn’t include forking) was created to increment whenever a csv or subdirectory was found.

**File contents:**

* testscan.c : contains the recursive forking functions that were tested to work.
* sorter.c : This function actual sorts the csv files - testscan.c was later merged with sorter.c to fork and sort .csv procs.
* sorter.h : contains the struct, headers, and sorter functions to be called by sorter.c
* mergesort.c : contains the mergesort algorithm used for sorting.
* movie\_metadata.csv : contains the movie file desired to be sorted
* -sorted-movie\_metadata.csv : contains the sorted movie file after merge sort based on movie\_title