Yitzy Dier, Jason Gaughan
CS214, Asst1
Documentation

scannerCSVsorter.h

Here are declared all helper methods employed by scannerCSVsorter.c. We also have the struct we use to store movie data from an individual row of a CSV in a doubly-linked list. For readability of the code, we define STDIN, STDOUT, and STDERR as the ints 0, 1, and 2 respectively, as well as introduce the traditional boolean enum. We also declare several extern variables explained below:

type	name	description
extern char **	fields	Array of column headers in a CSV
extern int	numfields	Number of columns in a CSV
extern int	sortColumn	Index in fields of the desired column on which to sort
extern int	dbSize	Number of rows in a CSV
extern char *	searchPath	Filepath to the current directory being searched
extern char *	savePath	Filepath to the desired save directory

scannerCSVsorter.c

call as SimpleCSVSorter -c header_name [-d startingDirectory] [-o outputDirectory]

METHODS
Methods listed in order of appearance

name	arguments	description
int main	int argc, char* argv[]	Checks arguments for header_name, startingDirectory, and ouputDirectory. If no syntactical errors, runs start()
int fileHandler	char *path, char *name	Forks process to determine if file name in directory path is a CSV. If CSV is found, it is opened and sorted along the sortColumn if possible, using mergeSort. Returns child PID, or -1 if failure
int dirHandler	char *path	Forks process to scan directory path for more files. Returns PID of child, -1 if failed, or (-1 – number_of_children) if the process is itself a child.
int scanDir	char *path	Called by start and dirHandler to scan directory path. Calls dirHandler upon finding a directory, calls fileHandler upon finding a file. Keeps track of number of calls and returns number of children it creates
int start		Called by main, calls scanDir on startingDirectory or on the current directory if none is specified. Returns number of

		children created, as counted by scanDir
void trim	char **strPtr	Removes all leading and trailing spaces from entries in string array strPtr
int readLine	FILE *csv, char **buffer	Reads line from csv until \n or EOF, using comma as delimiter for strings. Returns number of strings read and placed into buffer if at new line, or -1 if at EOF.
int setSortColumn	char *header	Finds index of header in fields, stores the value in sortColumn
int getFields	FILE *csv	Calls readLine on csv to populate fields, returns -1 if readLine does as well
int dbBuilder	FILE *csv, Movie **headPtr	Calls getFields to populate fields, calls setSortColumn on the sortString. Populates a doubly-linked list of Movie nodes with the data from fields. Returns -1 on failure to populate fields or set sortColumn
int saveCSV	Movie *head, char *path	Renames the file at path to “[path]-sorted-.csv” and populates it with its original data, but sorted, as stored in the doubly-linked list pointed to by head
int printCSV	Movie *head	Prints to STDOUT the doubly-linked list pointed to by head
void end		Frees searchDir and saveDir

mergesort.c

This code implements mergeSort in place on a doubly-linked list using the following three methods:

name	arguments	description
int mergeSort	Movie **hPtr	Sorts the doubly-linked list of Movie nodes whose head is pointed to by *hPtr by calling getMiddle on the list and recursively running mergeSort on the two halves. Then calls mergeSortMerge on the two halves to merge them together
Movie *mergeSortMerge	Movie *left, Movie *right	Merges two doubly-linked lists and returns a pointer to the first node
Movie *getMiddle	Movie *h	Returns a pointer to the middle node of a doubly-linked list whose head is pointed to by h

OPTIONS

When run from command line, the following flags my be used:

-c header_name: mandatory, specifies the name of the column on which the program will sort items

-d startingDirectory: specifies a directory for the code to begin executing in. Defaults to the current directory

-o outputDirectory: specifies a directory in which to ouput sorted files. Defaults to renaming and overwriting files where they are