CS351 Lab EC

Extra Credit

Instructions:

- *Assigned date: Saturday October 24th, 2020*
- *Due date: 11:59PM on Wednesday October 28th, 2020*
- *Maximum Points: 32*
- *This lab must be done individually*
- *Please post your questions to the Piazza forum*
- *Only a softcopy submission is required; it will automatically be collected through GIT at the deadline; email confirmation will be sent to your HAWK email address; submissions will be graded based on the collected submissions at the deadline; late submission will not be accepted*

1 Your Assignment

1. (10 points) Hash:
   Implement a string hash computation that computes a hash on the 1st command line argument (should support up to 2047 character long argument without spaces). You must implement 2 functions, including main() and hash(). The hash function should compute the hash as follows: take the ASCII decimal value of each character, and add all of the values together to get the 32-bit hash value. This hash value should be printed to the screen. The function declarations are included, and they must be followed exactly to receive full points. All your code should be contained in the hash.c file, and the Makefile.

   Usage:
   ./hash <string>

   For example, if you type:
   ./hash awesome
   OUTPUT: 753

   ./hash cs351
   OUTPUT: 367

2. (10 points) Bubble Sort:
   Implement a bubble sort program that will read from a file "pp2.txt" from the current directory a list of intergers (10 numbers to be exact), and the sort them, and print them to the screen. You can use redirection to read data from the file through standard input, as opposed to reading the data from the file with the read API (similar to Lab #1). You can assume the input data will only have 10 numbers in the file, and that the numbers will be valid integers. You must implement 3 functions, including main(), bubblesort(), and swap(). The function declarations are included, and they must be followed exactly to receive full points. All your code should be contained in the bubblesort.c file, and the Makefile.

   Usage:
   ./sort

   For example, if you type:
   ./sort
   INPUT: 1 3 2 8 4 9 8 3 10 14
   OUTPUT: 1 2 3 3 4 8 8 9 10 14

3. (10 points) Parallel:
   Implement the parallel utility that has 2 or more command line arguments:
   ./parallel <processes> <cmd> <args>

   For example, if you type:
   ./parallel 4 sleep 10
   OUTPUT: SUCCESS in running 4 parallel sleep tasks

   Implement the parallel utility that has 2 or more command line arguments. This will execute 4 copies of the sleep process for 10 seconds. Your program should allow an arbitrary number of command line arguments for the process to be run in parallel. The parent process should wait for all child processes to finish before exiting back to the shell. If the format of the command is not recognized (e.g. the first argument is not an integer), an error can be displayed. All your code should be contained in the parallel.c file, and the Makefile. You should handle the case that the program to be executed is succesfully executed, and the case where the program fails to execute.

2 What you will submit

When you have finished implementing the complete assignment as described above, you should submit your solution to your private git repository. Each program must work correctly and be detailed in-line documented. Your code must compile (with a Makefile), and must compile and run correctly with a variety of test cases (not provided) on fourier. To submit your work, simply commit all your changes to the Makefile (2 points), hash.c (10 points), bubblesort.c (10 points), and parallel.c (10 points) files and push your work to Github. You can find a git cheat sheet here: https://www.git-tower.com/blog/git-cheat-sheet/. Your solution will be collected automatically at the deadline. You cannot submit this assignment late. This assignment is optional and must be completed on-time. The extra credit points will be added to the programming assignment overall score. There is no need to submit anything on BB for this assignment. If you cannot access your repository contact the TAs.

Grades for late programs will be 0.