CSE 3320 Operating Systems
Fall 2019, © DL, UTA, 2019

Programming Assignment 2
Processes (simple), IPC, and threads
Due: On Canvas

Goal: Get experience creating and running concurrent processes and threads.

Part 1:

Description: You will create processes, run processes in parallel, and pass information between processes.
The data to be processed are lines of CSV separated values, we wish to sort these.

1. Get one of these data sets:
   http://earthquake.usgs.gov/earthquakes/feed/v1.0/csv.php (all quakes last 30 days) or
   http://www-odi.nhtsa.dot.gov/downloads/flatfiles.cfm and get the FLAT_RCL file
   (zip) which are all vehicle recalls in the US (the RCL.txt is the "meta"-data (schema) description of the data) or
   If you sort the earthquake data, sort by latitude, in ascending order, if you sort the recalls sort by year of recall and manufacturer (first year, then manufacture name).
2. Either: sort by hand(not a good idea), or write a two loop (bubble or insertion) sort program to sort the data.
   (Do not use a "better" (faster) sorting method, but if you really want to, use much, much bigger data.)
3. "Instrument" your program (time it).
4. Create a program that will, in turn, run multiple processes "concurrently" using "fork( )" and "exec( )"
   (there are several variants of exec (execl, execv, etc.)
   Please do not use "threads" (yet).
5. Do the sort, again, in parallel for 2 concurrent processes, then 4, and finally 10 processes.
6. Instrument those sorts (like above).
7. (?) How will you pass data (parts of the array) to each process (IPC)?
   (Files, pipes, shared memory, message queues?)
8. You may (not required) to use a menu to select number of processes, size of data, etc.

Part 2:

Description: You will create a process and threads, run in parallel, and pass information between threads. You will use synchronization operations to avoid overwriting shared resources.

As in Assignment above:

1. Use the same data set, as assignment 2.
2. Either: sort by hand, or write a two loop (bubble or insertion) sort program to sort the numbers in ascending order.
3. "Instrument" your program (time it).
4. Create a program that will, in turn, run multiple threads "concurrently" using a kernel level threading system. (there are several options: Pthreads, Java, C/C++)
5. Do the sort, again, in parallel for 2 concurrent threads, then 4, and 10 threads.
6. Instrument those sorts (above).
7. (?) How will you pass data (parts of the array) to each thread? (How will you synchronize sharing resources such as memory?)
8. You may (not required) to use a menu to select number of threads, size of data, etc.

Hint:

Lawrence Livermore Lab thread tutorial: https://computing.llnl.gov/tutorials/pthreads/
includes thread creation, management, and using mutexes