CS 366
Assignment #7
Threads!
Due 4/28/17, in class

Goal

Write a multi-threaded program. A thread is an independent locus of control within a process (for the time being, a process is a program in execution). Thus if you have several things to do and several CPUs, using multiple threads can happen at the same time and thus speed things up!

Problem Statement

Many different sorting algorithms have been devised over the years. In addition to their complexity (e.g., $O(N^2)$ versus $O(N\log N)$), they differ in other ways. For example, a sorting algorithm is stable if two objects with equal keys appear in the same order in sorted output as they appear in the unsorted input. One of Mergesort’s advantages is that it is very easy to parallelize. One approach is to have different threads sort different chunks of the input data. The sorted chunks can then be merged together.

Your task is to write a multi-threaded Mergesort that takes as command line arguments an element count and a number of threads to use. Your program should create an array of random integers and then sort it. The element count will be given as a power of two and you can assume that element count is evenly divisible by the number of threads. For example, the command `mergesort 10 4` would create an array of $2^{10}$ elements and then sort it using 4 threads. Each thread would Mergesort a quarter of the data and then the four sorted chunks would be merged together. The only “output” of interest is the time taken (as printed by the code found in the repo). Finally, run your code twice in one of two different modes: checking-mode, which will use memwatch and `qsort` to help ensure correct operation, and performance-mode, in which memwatch and `qsort` are not used so that the true run time can be gathered.

What to hand in  (Please no .docx files.)

1. A well-formatted 2-up printout of your source code. You must use `a2ps` after removing all the tabs from your code. Indent code 2 or 4 spaces at most.

2. A GitHub repo that includes (you must use these names as the grading script will assume their use!)
   - `README.md` (this should include your analysis, design, and test-plan),
   - your source code (at least `main.c`, `mergesort.c`, and `mergesort.h`), and
   - `Makefile` (where `make` will build your code in performance mode, and `make checking` will build your code using memwatch (reporting zero anomalies) and `qsort`).

Assignment Notes

• I expect to pull your code, run `make`, and then run my test script.
• The output of `git log` will again factor into your grade.
• Here is the git classroom invitation
  `https://classroom.github.com/assignment-invitations/f32248b5984b9154cc68579ceaea72c4`
• Want some more fun? Rather than sorting integers using hard coded thread assignments, support the sorting of arbitrary elements and the arbitrary chunking of the data where each chuck is sorted and then merged with every larger neighboring sorted chunks.