CSC 543 Multiprocessing & Concurrent Programming, Spring 2023 First Day Handout

https://faculty.kutztown.edu/parson/spring2023/CSC543Spring2023.html

Monday 6-8:50 PM, http://faculty.kutztown.edu/parson This course is multi-modal, meaning you can attend in person in Old Main 158 or via Zoom at class time.
Class-time Zoom for CSC543: See D2L Course CSC543 -> Content -> Overview for the link.
IF you don't want to be recorded or are a minor, use PRIVATE ZOOM CHAT to me for questions.
Dr. Dale E. Parson, parson@kutztown.edu, Office hours: https://kutztown.zoom.us/j/94322223872
Office Hours Monday 3-5 PM, Tuesday 3-4 PM, Friday (Zoom only) 3-5 PM, or by appt.
Monday & Tuesday office hours are either Zoom using the above link or at Old Main 260.

This course explores the concepts and practices of creating software that makes effective use of modern multiple-processor computers. Emphasis is on partitioning program code data for safe and efficient execution on multiple processors that share machine resources such as memory. Lab exercises include construction, execution, and benchmarking of multithreaded programs on several multicore, multithreaded computers.

Textbook: *Java Concurrency in Practice*, Brian Goetz, et. al., 2006. Get it!

Grading (A = 92:100, A- = 90:91, B+ = 87:89, B = 80:86, C+ = 77:79, C = 70:76, F = 0:69)

Final project + report	20% of grade (a student project and report)
Programs	80% divided equally among 4 programming assignments

Programming project assignment grading criteria

Please follow my detailed requirements in assignment handouts.
Test everything before turning it in via **make turnitin** after running **make test** a final time.
When you think you are finished, read the requirements to avoid missing anything.
Test it after any changes.
I will deduct points for missing documentation comments required in the handout.
Team project grades will include peer review points from your teammates.

The academic integrity policy is at http://cs.kutztown.edu/pdfs/AcademicIntegrityPolicy.pdf

Your first reading assignment is to read the above policy statement.
You may openly discuss ideas, algorithms, pitfalls, and the use of programming tools.
You may not share code, test drivers or test data except within groups for group projects.
Group projects, when assigned, have documented partitioning of student responsibilities.

Class attendance is not graded, but I will be teaching using data sources and concepts both inside and outside the scope of the textbook. You are responsible for all material covered in class, including technical information, coding standards and conventions, verbal specification of assignments, and your questions about topics that are not clear to you. Please, there should be no classroom conversations, cell phones, text messaging, eating, sleeping, obscenities, smoking (tobacco or artificial), vaping, listening to music or other disruptions of the class. I will deduct 5% from an assignment for each infraction.

If you have already disclosed a disability to the Disability Services Office (215 Stratton Administration Building) and are seeking accommodations, please feel free to speak with me privately so that I may assist you. If you have an injury sustained during military service including PTSD or TBI, you are also eligible for accommodations under the ADA and should contact the Disability Services Office.

If you have preferred pronouns for yourself, or a name that differs from the MyKU roster, please let me know.

We will be using in-person attendance OR Zoom for remote attendance during class time. Recorded archives of class sessions will be available. We will go over this in the first class.

Week	Text	Lecture Topics
1	handouts	Shared memory multiprocessor servers and CPU-NUMA-memory latencies.
2	Ch 1-3, 13	Critical sections, locking and condition variables, sharing objects
3	Ch 15-16, A	Volatile and atomic variables, non-blocking synchronization, memory model
4	Ch 4	Composing objects – thread-safe object-oriented composition
5	Ch 5	Building blocks, library collection classes, java.util.concurrent packages
6	Ch 6	Task execution, the Executor framework, finding exploitable parallelism
7	Ch 7	Cancellation and shutdown, daemon threads
8	review	Introduction **to** thread pools, benchmark projects
9	Ch 8	Thread pools, parallelizing recursive algorithms
10	Ch 9	Threading issues for GUI applications and RMI distribution, example code
11	Ch 10-11	Avoiding liveness hazards, performance scalability, benchmarks
12	Ch 12	Testing, benchmarks, introduction to C++0X library classes.
13		SIMD applications of GPUs, MIMD DSPs and network processors
14		Student talks, 20-30 minutes each, on benchmarks, 20% of course
15		Student talks, 20-30 minutes each, on benchmarks, 20% of course

There are five projects in the plan, with project sequence to be determined.

1. Design and implement a producer-consumer *pipeline* using basic synchronization mechanisms.

2. Design and implement a *parallel* implementation of a CPU-intensive search algorithm with CylicBarrier synchronization.

3. Design and implement a *parallel* implementation of a CPU-intensive search algorithm using a minimal-locking dataflow approach.

4. Design and implement a simple *server* application that uses a *worker thread pool*, with emphasis on minimal-latency response to client requests.

5. Project 5 is a custom individual student project. I will consider a proposal for a more advanced project with a team of 2 students.