




## Lecture 22: Parallelism & Concurrency

Slides adapted from Dan Grossman

### + Today



- Reading
  - A Sophomoric Introduction to Shared-Memory Parallelism and Concurrency (S&C) Section 2
- Objectives
  - Weekly lab and assignment
  - Introduce parallelism and concurrency
- Announcements
  - Weekly programming assignment can be done with partner!

## + This week's assignment and lab

- Create a `CompleteBinaryTree<E>` class
  - Values are added and removed in such a way that completeness is preserved
  - Should be a straightforward assignment
- Experiments with binary search trees (BST)
  - Method to insert value into existing BST
  - Compute height of BST of randomly generated integers

## + Sequential Programming

- Almost everything you're taught involves sequential programming
  - 1980 – 2005: computers twice as fast every 18 months to 2 years!
- Limitations of sequential programming
  - Nobody knows how to continue this trend
  - Computers now have multiple cores – how to take advantage?
  - Computation (e.g. searching, sorting) involves petabytes of data
- Moving beyond sequential programming
  - Parallelism
  - Concurrency

## + What does it look like?

- What does it look like to move beyond sequential programming?
  - Programming
    - Divide work among **threads of execution** and coordinate (**synchronize**) among them
  - Algorithms
    - How can parallel activity provide speed-up (more **throughput**, i.e. more work done per unit time)
  - Data structures
    - May need to support **concurrent access** (multiple threads operating on data at the same time)

## + Parallelism vs. Concurrency

- Sequential programming:
  - One thing happens at a time
- Parallelism:
  - Use additional computational resources to produce an answer faster
- Concurrency
  - Correctly and efficiently controlling access by multiple threads to shared resources

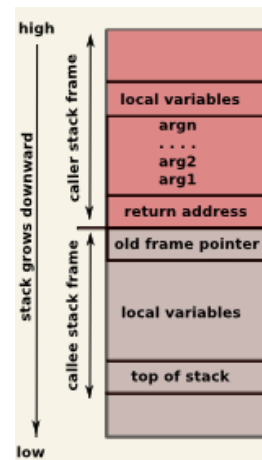
## + Cooking Analogy

- Sequential programming
  - One cook performing each step of a recipe
  - Each step finished before next one started
- Parallelism:
  - Extra cooks (or equipment) to finish faster
  - At some point, extra hands don't help anymore
- Concurrency:
  - Lots of cooks but only one oven!
  - Coordinate access to oven so no burning or crowding



## + Sequential Programming

- When running, a sequential program has:
  - One call stack
  - Each stack frame holding local variables
  - One program counter pointing to memory location of currently executing instruction
  - Static fields of classes
  - Objects created by calling new stored on the "heap" (not related to heap data structure)



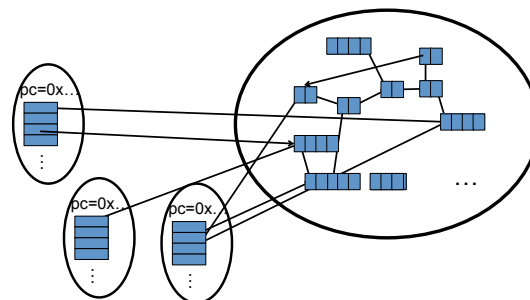
## + Explicit Threads with Shared Memory

- Threads
  - The smallest unit of execution – similar to running a sequential program
  - Each thread has its own call stack, program counter, local variables
  - Threads share static fields and objects
  - Threads communicate by writing/reading to same objects
  - To communicate, write somewhere another thread can read

## + Shared Memory

Threads with own unshared call stack and program counter. Local variables are numbers/null or heap references

Heap for all objects and static fields



## + Summary

- Sequential programming
  - Limitations of this approach
- Parallelism versus concurrency
- Explicit threads with shared memory
  - Alternative models in text
- Thread is a single unit of execution
  - Separate call stacks, program counter, local variable
  - Shared static fields and objects