Lecture 7.2: OS Abstractions Threads

EN 600.320/420

Instructor: Randal Burns

19 February 2018

Department of Computer Science, *Johns Hopkins University*

What's a thread?

- Thread = concurrent execution unit within a process
  - Threads share memory (entire virtual address space)

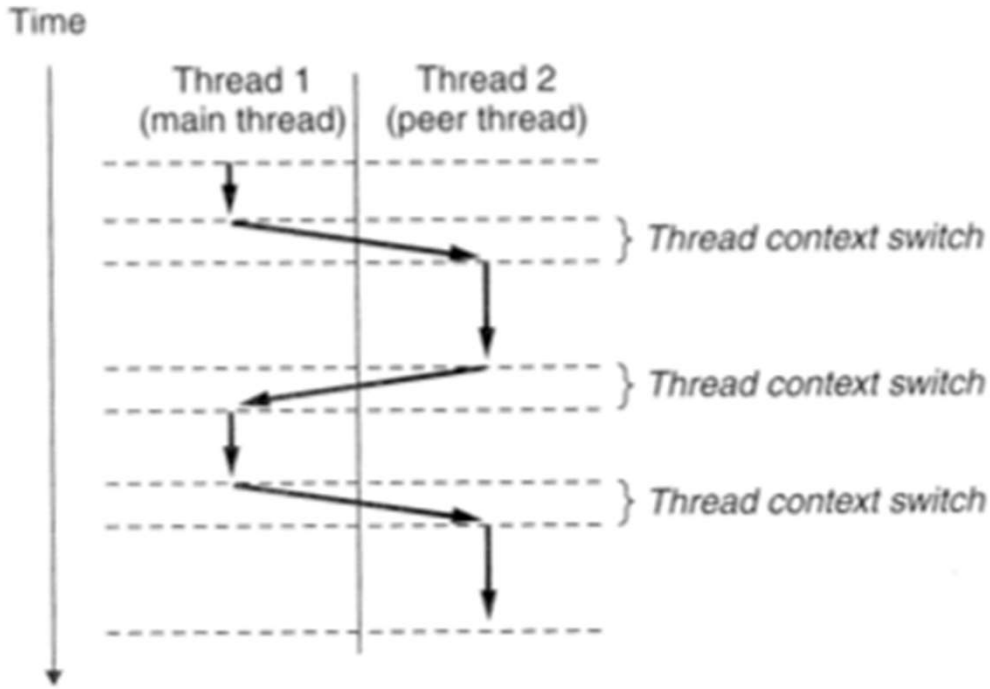
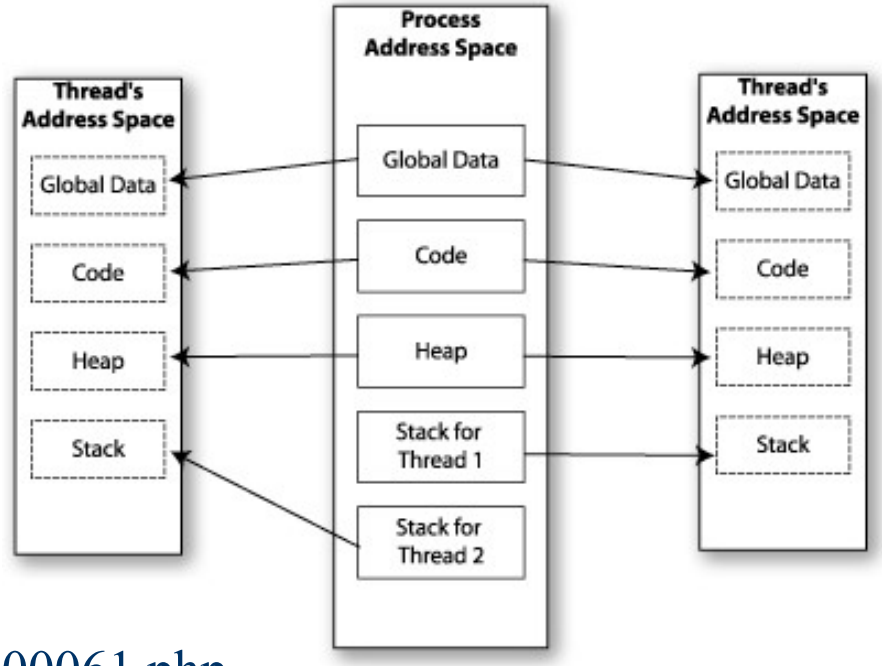


Figure 13.12 **Concurrent thread execution.**

Thread Address Space

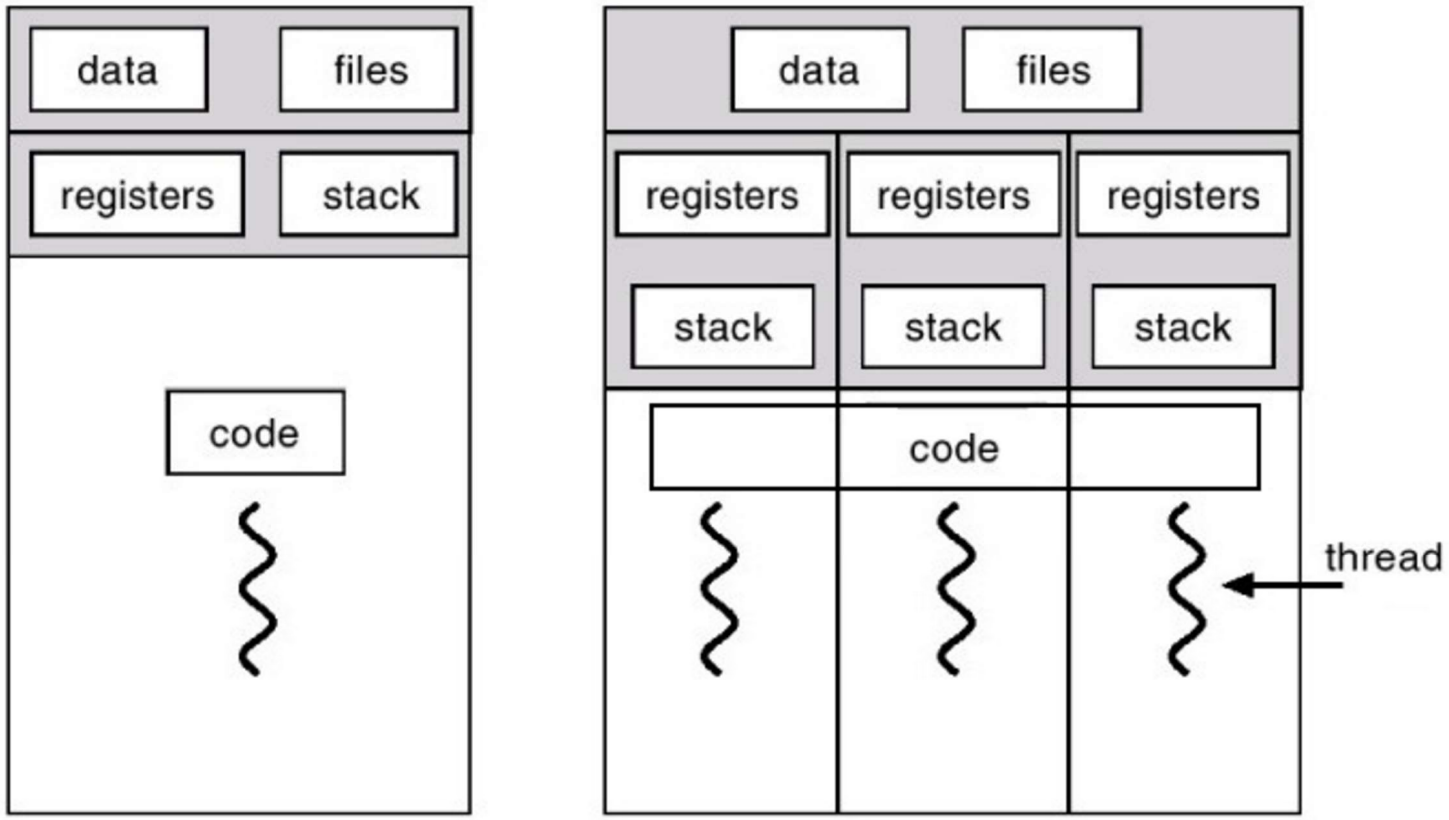
- Lightweight context
  - thread identifier, stack, registers
- Shared heap, libraries (this means all new'ed objects)
- Data sharing via shared memory
  - No IPC



http://cocoadevcentral.com/articles/000061.php

Threads and Processes

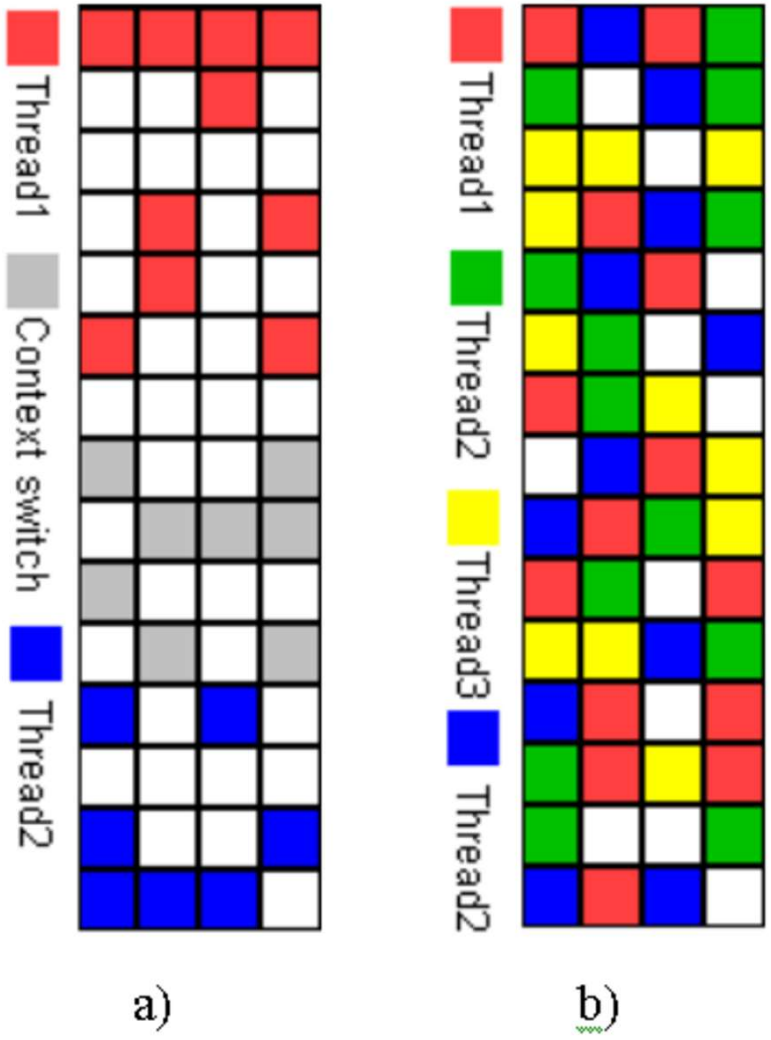
- Every process has a at least one thread



http://www.cs.miami.edu/home/visser/Courses/CSC322-09S/Content/UNIXProgramming/UNIXThreads.shtml

Simultaneous Multithreading

- Simultaneous multithreading
  - Run multiple threads on each core each cycle
- Hyperthreading: SMT for Intel
  - Processor and OS advertise two threads per core
  - Simple programming interface
  - Don't get full performance out of both threads: 15-30% speedup



Threads and Parallelism

- For embarrassingly parallel programs
  - Linear speedup with each core
- Incremental benefits from multithreading
  - Simultaneous multi-threading (+15-30%)
  - Overlap cache stalls with the execution of other threads
  - Decrease cycles per instruction (<1 for some workloads)
- How many threads to use is confusing

Advantages/Disadvantages of Threads

- Shared variables!
- Easy communication between execution contexts
  - Multiple threads coordinate their actions and share data through reading and writing shared variables
- Hidden dependencies hard to debug
  - Variables may be updated by other processes: change from serial mindset
- OOP to the rescue
  - Objects provide encapsulation in support of threading
  - Classes control access to shared data via synchronization, volatile, and atomic language features