Threads: An Instant Primer

ECE 353S: **Systems Software**

Baochun Li
University of Toronto

The Process Model Revisited

The process model is based on two concepts —

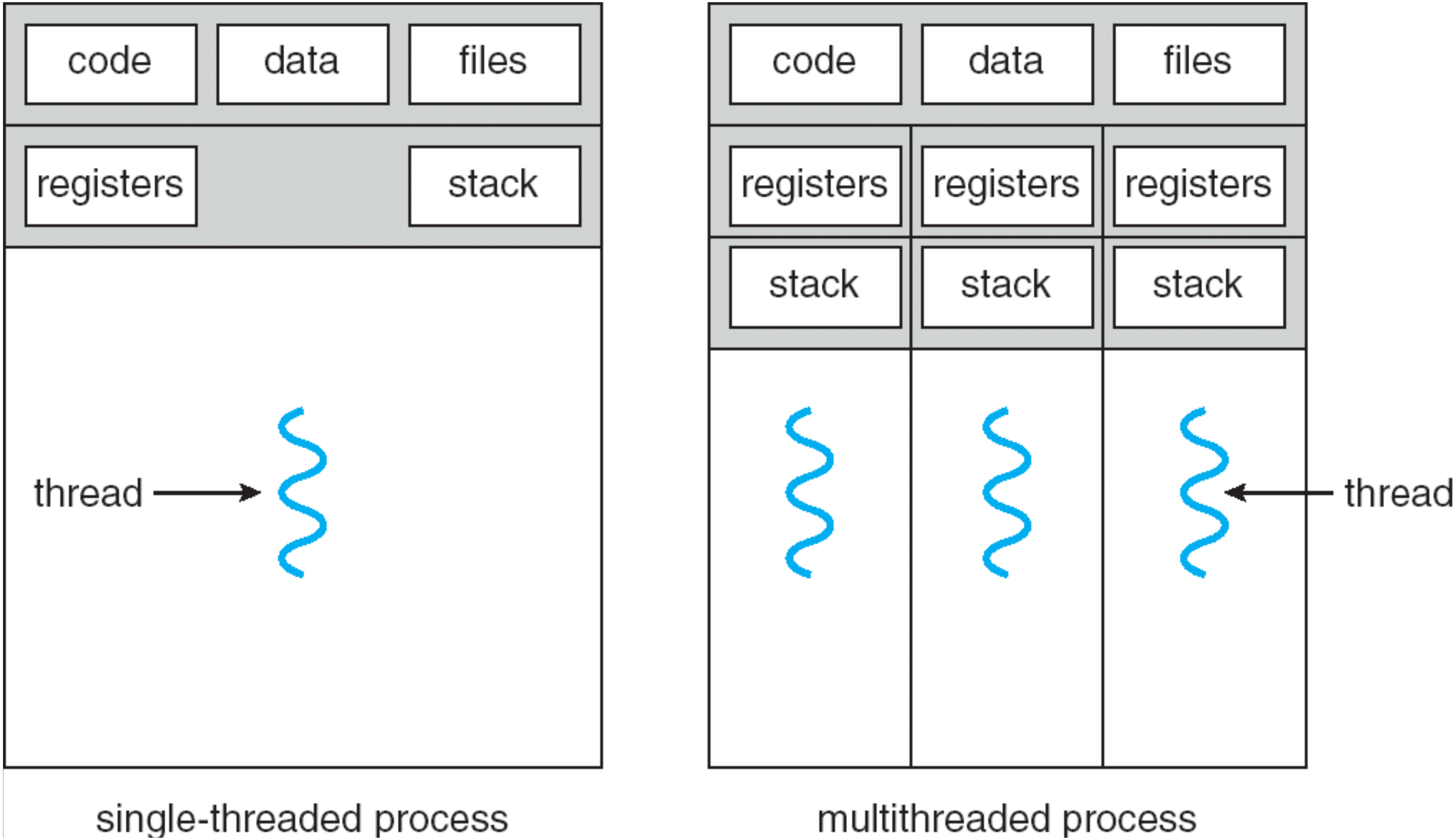
- grouping of resources
  - program text and data sections
  - open files
- execution — a “thread” of control
  - program counter, stack pointer, and registers

They can (and should) be treated separately

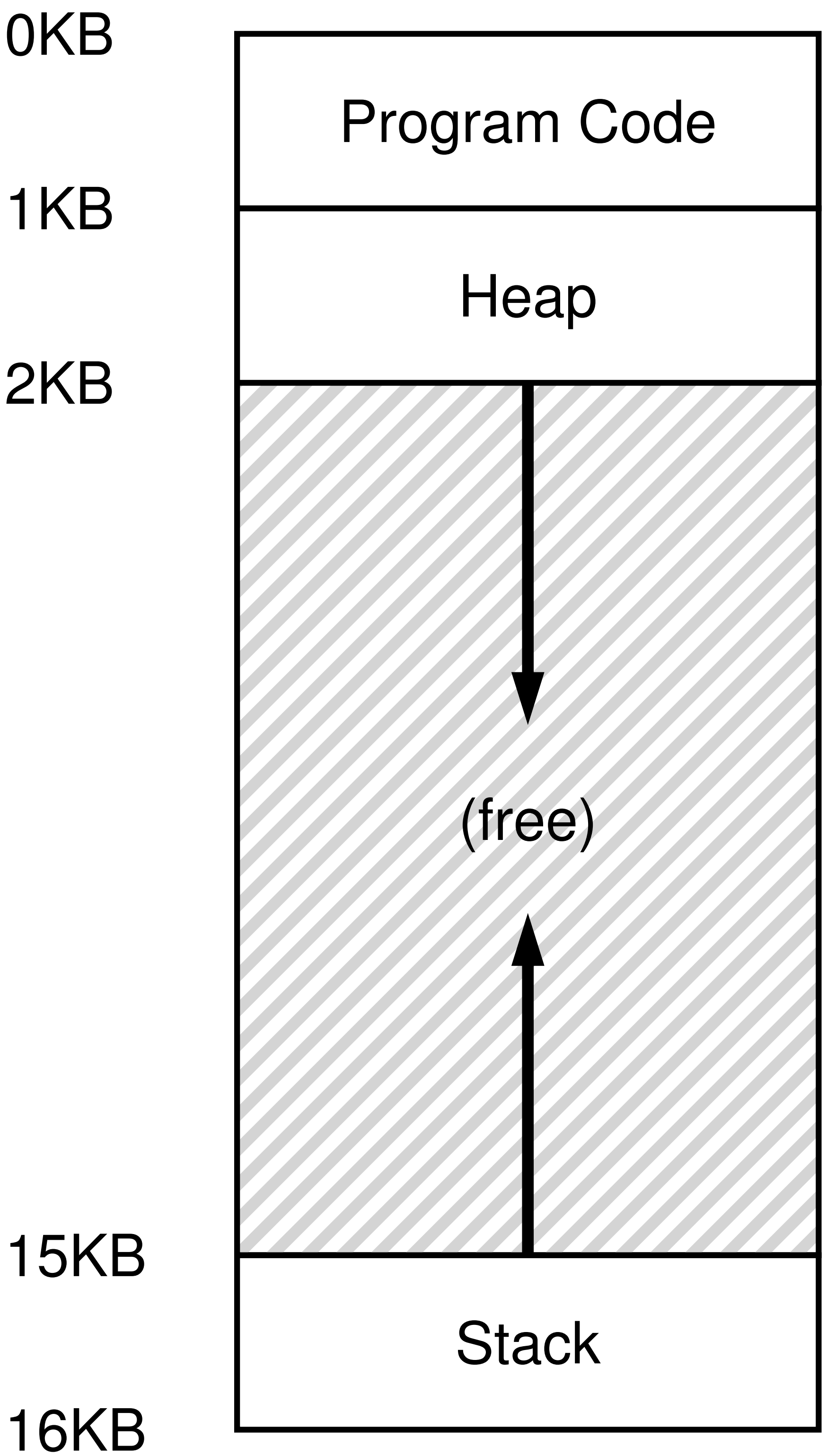
- Analogy: spouses share the same house, mortgage, and bank account, but they can enjoy different lives every day

Now we have threads

Single-threaded vs. multithreaded processes



Threads in a virtual address space

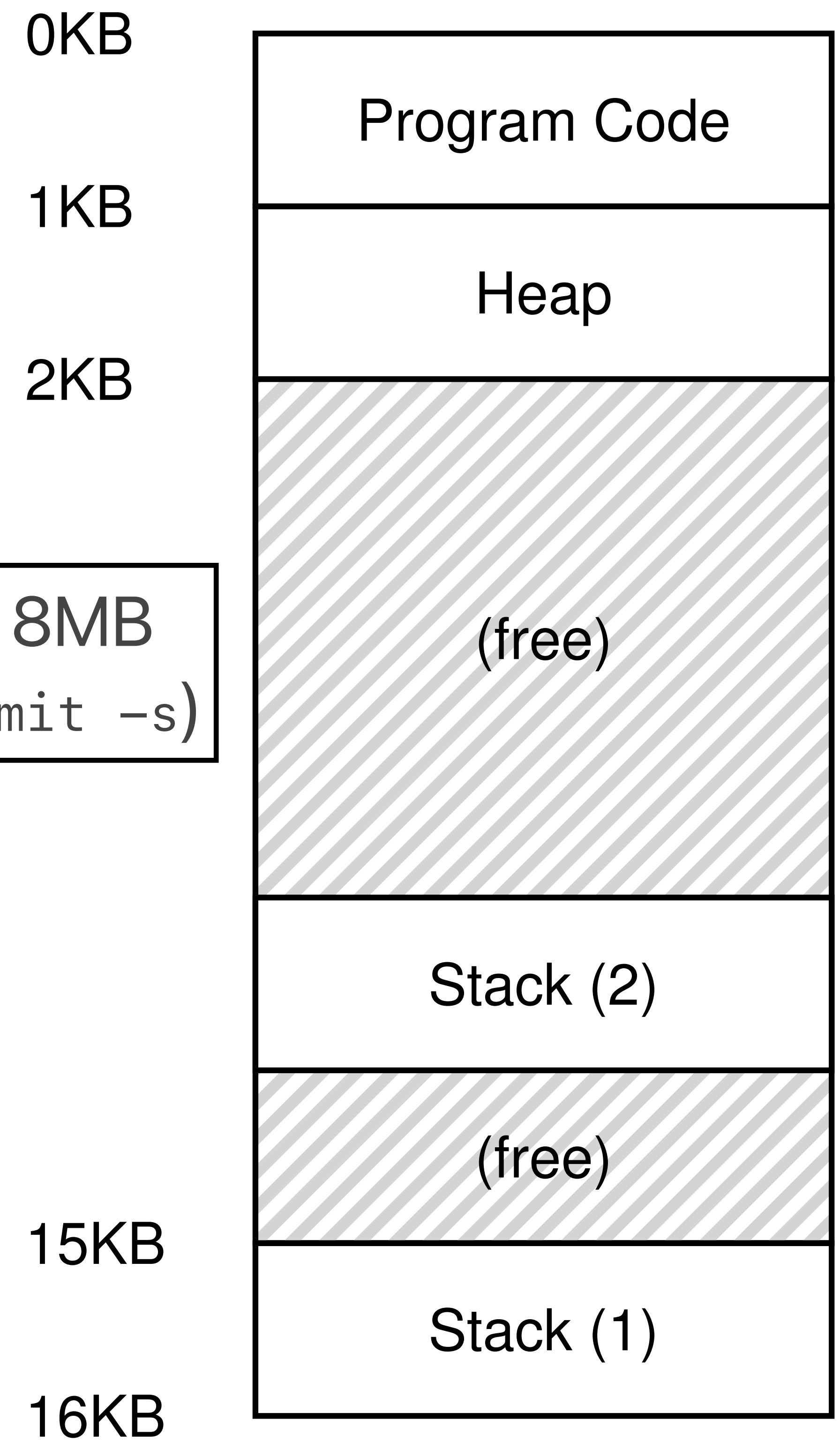


the code segment:
where instructions live

the heap segment:
contains malloc'd data
dynamic data structures
(it grows downward)

Stack size per thread is 8MB by default in macOS (`ulimit -s`)

(it grows upward)
the stack segment:
contains local variables
arguments to routines,
return values, etc.



Why Do We Need Threads — Intuition

Sometimes a program needs to do multiple tasks concurrently

- Consider a word processor that needs to do automated backup while you are typing
- You cannot achieve this with multiple processes, since the backup and typing are on the same document (address space)

Thread Scheduling States

The thread scheduler multiplexes threads on physical CPUs

The scheduler decides which thread to run based on the thread state

Thread states

- Running — thread is using processor
- Blocked — thread is waiting for input
- Ready — thread is ready to run
- Exited — thread has exited but not been destroyed

Thread scheduling primitives

yield() — Current thread yields the CPU

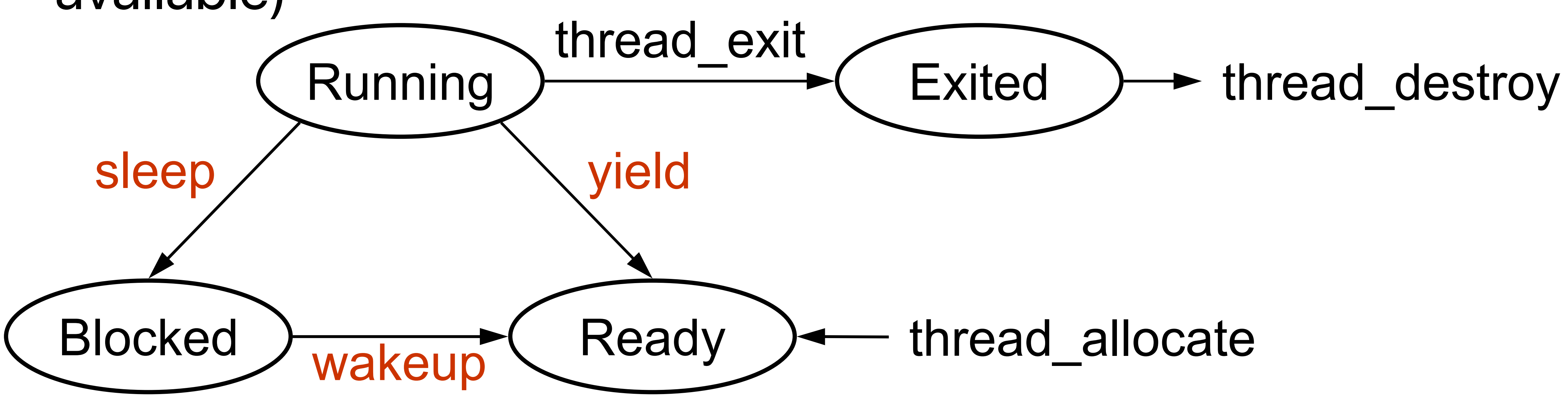
State change: Running -> Ready

sleep() — Current thread blocks for some reason

State change: Running -> Blocked (e.g., sending to full buffer)

wakeup() — Another thread wakes up a thread

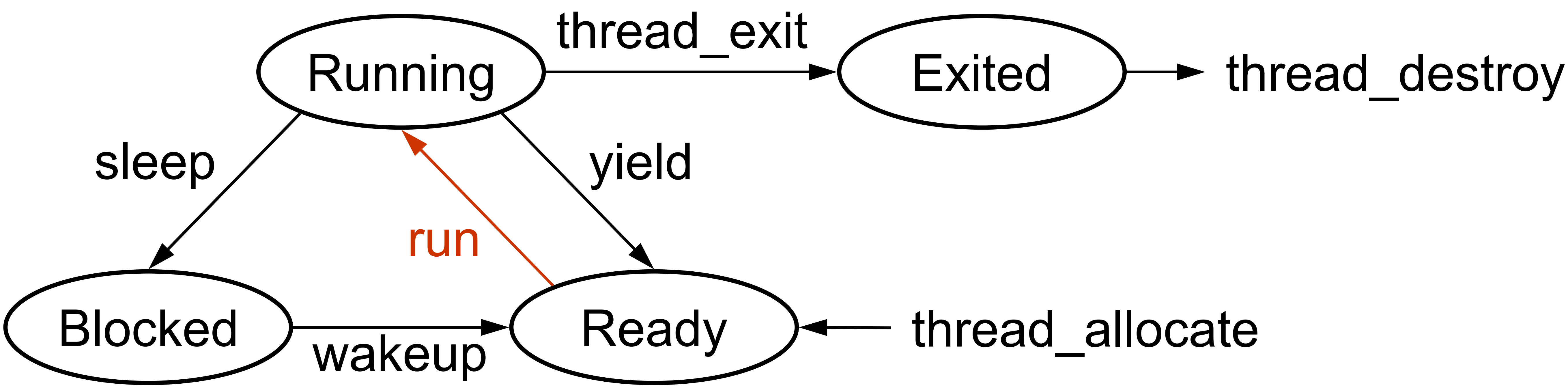
State change: Blocked -> Ready (e.g., buffer space becomes available)



The Thread Scheduler

Chooses another ready thread to run based on a scheduling policy

Runs as a result of scheduling primitives yield() and sleep()



What we've covered so far

Three Easy Pieces: Chapter 26.1 and 26.2 (Concurrency: An Introduction)

If you think debugging is hard, think twice before using threads —

- Next up: race conditions