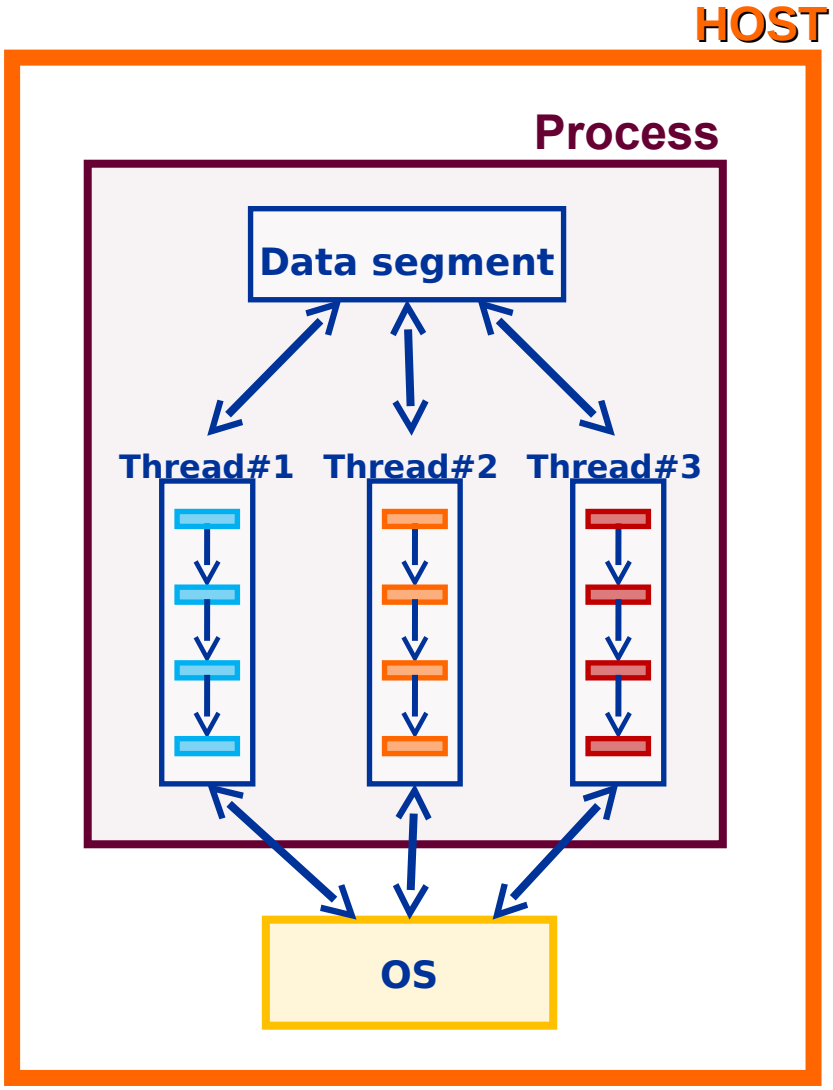
Section 5.4
Threads

1. Overview
2. Sharing data

5.4.1 Overview

- What is a *thread*?
  - it’s a single logical flow of control inside a process
- Thread management
  - threads can be created by the main thread or by peer threads
    - thread creation specifies a function for the thread to execute
  - threads run in parallel
    - they are scheduled automatically by the OS kernel
    - the context switch between threads is faster than between processes
- **coding example <p1>**

Overview (cont.)



Overview (cont.)

- Characteristics of threads
  - each thread has its own unique resources (*thread context*):
    - a thread id
    - a function call stack
    - a program counter
      - points to instruction currently executing
  - all threads within a process share the same:
    - address space
    - data segment
      - including the heap
    - code segment

- **`coding example <p2>`**

5.4.2 Sharing Data

- What areas of memory are shared between threads?
  - global memory
  - heap
- Problem
  - two threads modifying the same data at the same time
- **`coding example <p3>`**

Sharing Data (cont.)

- Solution:
  - protect the shared data
  - ensure that changes to the shared data are **atomic**
    - atomic operations are *non-divisible* operations
    - their time-slice on the CPU cannot be *preempted*
      - it cannot be interrupted by another thread or process
  - mechanisms to protect shared data:
    - semaphore
    - mutex

Semaphores

- What is a *semaphore*?
  - it's a variable used in only *two* atomic operations
    - *lock* operation
    - *unlock* operation
  - it's initialized with the number of threads that can lock it simultaneously
  - if it's locked, other threads trying to access it are blocked
  - once unlocked, *no guarantee* which thread gets unblocked

Semaphores (cont.)

- Operations on semaphores
  - wait operation
    - this is the *locking* operation
    - if the semaphore is non-zero, operation decrements the semaphore
    - if the semaphore is zero, the thread is blocked until it's non-zero
  - post operation
    - this is the *unlocking* operation
    - it increments the semaphore

Mutexes

- What is a *mutex*?
  - it’s a binary semaphore
    - its value is zero or one
  - only *one thread* can access it at a time
  - mutex == **mut**ual **ex**clusion
- **`coding examples <p4> and <p5>`**