Project 2
Non-Preemptive Scheduling

COS 318

- Design Review
  - Monday, October 8
  - Similar hours as Project 1

- Due Sunday, October 14 at midnight

General Suggestions

- Use an IDE
  - Eclipse
    - Built into lab machines
      - Help -> Install New Software...
      - Download a specific Eclipse package for C/C++ from eclipse.org
  - Others
- Start as soon as you can and get as much done as possible by design review time

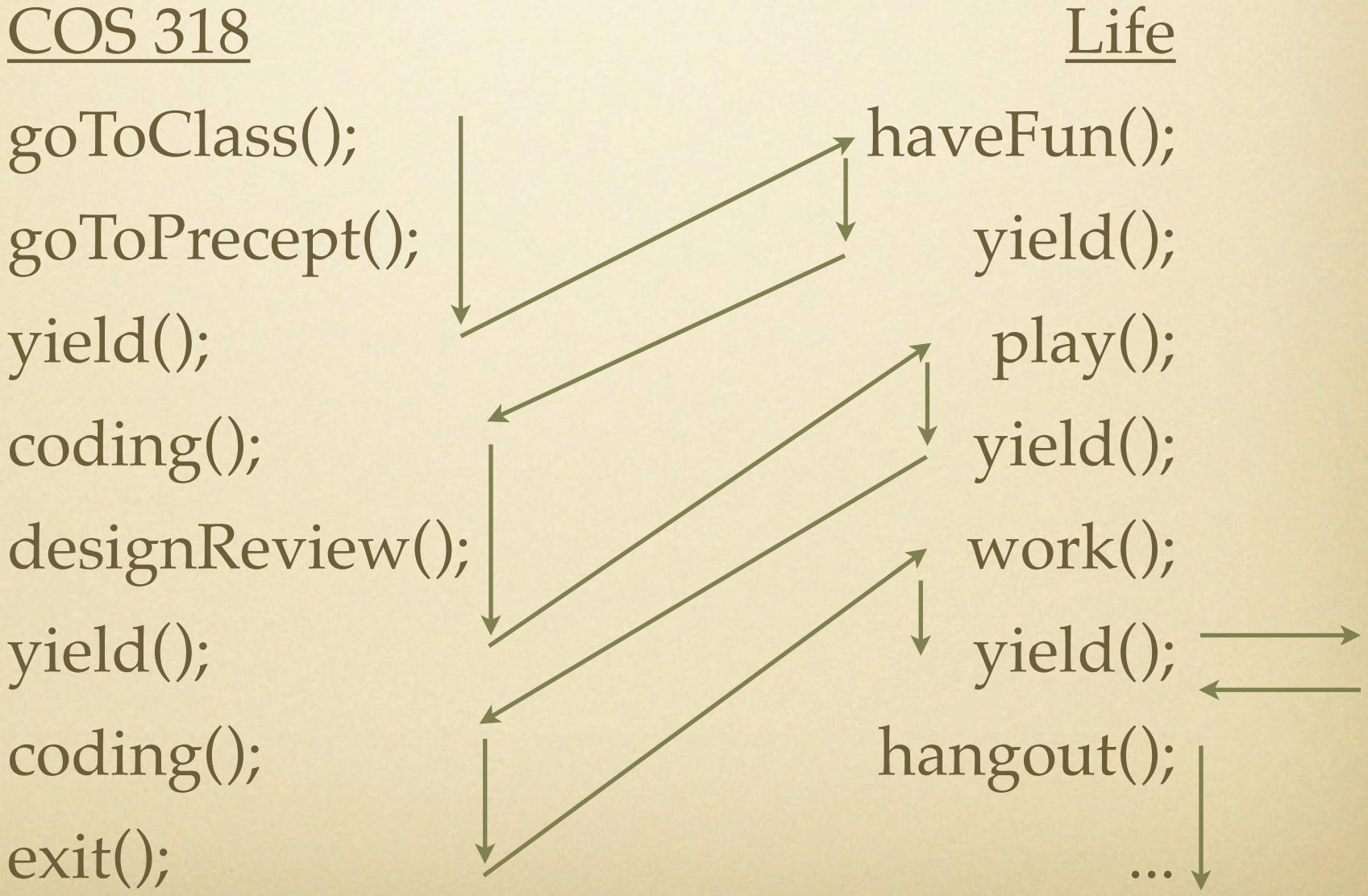
Overview

- Add multiprogramming to the kernel
  - Non-preemptive scheduler
    - 5 threads, 3 processes
      - Process Control Blocks
    - Context switching
      - Timing
- Mutual exclusion
  - Lock

Non-Preemptive

- What does it mean?
- yield & exit
  - do_yield() & do_exit() within the kernel (kernel threads can call these directly)
  - yield() & exit() for processes
    - dispatches a desire to call do_yield() or do_exit() to the kernel

Non-Preemptive Scheduling Example



What yield does

- When yield is called, the “context” of a task (thread or process) must be saved
- Process Control Block
  - What does it contain?
- Will be done in assembly
- Once the context is saved, the scheduler is run to pick a new task

Picking a New Task

- All tasks are waiting in a queue to be run
- Pick the next one from the front
- Restore it's state from the PCB
- Return to where the task was executing before

Mutual Exclusion

- lock_init(lock_t * l)
- lock_acquire(lock_t * l)
- lock_release(lock_t * l)