CS 3723 Operating Systems: Final Review

Instructor: Dr. Tongping Liu
Outline

- Threads
- Synchronization, critical sections and semaphores
- Thread Synchronization
Lecture-08: Threads

- Motivation and thread basics
  - Resources requirements: thread vs. process

- Thread implementations/Thread Model
  - User threads: e.g., Pthreads and Java threads
  - Kernel threads: e.g., Linux tasks
  - Map user- and kernel-level threads

- Threading Functions: thread creation/join/exit

- Other multithreaded issues

- Threads Memory Model
Important Points of Lecture-08

- Similarities and difference between threads and processes (C)
- Shared and separated resources of each thread (C)
- Pros and cons of thread-based designs (B)
- Three multithreading models, pros and cons (B)
- Thread creation (C)
- Thread Arguments Passing (A)
- Thread joins (C)
- Thread exits (A)
- Fork() in a multithreaded program (B)
Important Points of Lecture-08

- Signal handling in multithreading (A)
- Threads memory model (C)
- Instances of global, local, or local static variables (B)
- Shared variable analysis (B)
Lecture-09: Synchronization, Critical Sections and Semaphores

- Problems with concurrent access to shared data
  - Race condition and critical section
  - General structure for enforce critical section

- Software based solutions:
  - Simple solution
  - Peterson’s solution

- Hardware based solution
  - Disable interrupts
  - TestAndSet
  - Swap

- OS solution -- Semaphore
  - Using semaphores to solve real issues
Important Points of Lecture-09

- Race condition definitions and examples (C)
- Definition of synchronization (C)
- Critical section (C)
- General structures of critical section (C)
- Requirements of CS solutions (C)
- Mutual exclusion (C)
- Peterson’s solution: guarantee mutual exclusion (A)
- Issues of using disable Interrupt (A)
- Using TestAndSet to solve CS issues – Quiz10 (B)
- Using Swap() to solve CS issues – Quiz10 (B)
Important Points of Lecture-09

- Semaphore operations (C)
- Basic idea of semaphore (B)
- Counting and Binary Semaphore (C)
- Using semaphores to solve real issue: critical section, control the order, and producer-consumer problem (C)
Lecture-10: Thread Synchronization

- Pthread mutex
- Conditional variables
- Barrier
- Threading Issues
- High-level synchronization structure: Monitor
Important Points of Lecture-10

- Solve the CS Issue using mutex lock (C)
- Difference between binary semaphore and pthread mutex (B)
- Operations of condition variables (C)
- Barrier (B)
- Using mutex lock and conditional variable to solve shared-buffer (producer-consumer) issue (B)
- Deadlock issue in assignment4 (C)
- Monitors vs. semaphore (B)
- Explicit locking vs. implicit locking (Monitors) (A)
Overall

- Process: concept, memory model, scheduling, basic programming (fork(), wait())
- File system: file, file pointer, links, FDT, SFT, inode
- IPC: pipe, fifo, dup2
- Memory management: paging, page table, TLB, buddy, page replacement
- Threads: concept, memory model, difference with processes, threading model, basic programming
- Synchronization: hardware instruction, semaphore
- Thread-Based Synchronization: lock, conditional variables, barrier, synchronization issue
Overall Distribution

- Process, Scheduling and IO: ~20%
- Memory Management: ~ 30%
- Threads & Synchronizations: ~ 50%

- Total: 320 points