Midterm II - reivew

Instructor: Dr. Tongping Liu
The University of Texas at San Antonio

Outline of Lecture-04

- Motivation and thread basics
  - Resources requirements: thread vs. process
- Thread implementations
  - User threads: e.g., Pthreads and Java threads
  - Kernel threads: e.g., Linux tasks
  - Map user- and kernel-level threads
  - Lightweight process and scheduler activation
- Other issues with threads: process creation and signals etc.
- Threaded programs
  - Thread pool
  - Performance vs. number of threads vs. CPUs and I/Os

Outline of Lecture-05

- Problems with concurrent access to shared data
  - Race condition and critical section
  - General structure for enforce critical section
- Synchronization mechanism
  - Hardware supported instructions: e.g., TestAndSet
  - Software solution: e.g., semaphore
- Classical Synchronization Problems
- High-level synchronization structure: Monitor
- Case study for synchronization
  - Pthread library: mutex and conditional variables
  - Java inherit monitor and conditional variable

Important Points

- Difference between threads and processes (B)
- Pros and cons of using threads (B)
- Three multithreading models, pros and cons (B)
- Thread creation (B)
- Thread joins (B)
- Thread exits (A)
- Threads memory model (B)
- Instances of global, local, or local static variables (B)
- Thread pool advantages (A)
- Performance of threaded programs (A)
Important Points

- Race condition definitions and examples (B)
- Critical section (B)
- General structures of critical section (B)
- General requirements of solving CS problems (A)
- Mutual exclusion (B)
- Using TestAndSet and Swap to solve CS issues (B)
- Semaphore operations (B)
- Counting and Binary Semaphore (B)
- Using semaphores to solve any synchronization problem (B)

Problems of using semaphores (A)

Difference between semaphores and monitors (B)

Pthread mutex and conditional variables (B)

Difference between binary semaphore and pthread mutex (B)

Usage of mutex and conditional variables (B)

Barrier (A)