Midterm Review

CS 416: Operating Systems Design
Department of Computer Science
Rutgers University
Reminders

- Write name on all answer books.

- Closed book. Closed notes. No calculators, cell phones, laptops, etc.

- No questions during the exam. If in doubt, make an assumption and write it down.

- Be concise. Write just enough to answer questions.

- Cheating will be punished severely!
Architecture

- Caching? How does it work, Types of caches, types of misses
- Exceptions, traps, interrupts, signals (How are they created, what happens under each of these situations)
Processes and Threads

- Stack. Activation records. What is stored there?
- Heap. What is stored there?
- Address space. What is it?
- Process context. Thread context. What do they comprise?
- PCB. TCB. Where are they stored? What do they store?
- Fork(), exec()
- Context switching. Steps involved in Context Switching
- Thread states. Transitions between states.
- User-level and kernel-level threads. Pros and cons?
Synchronization

- **Critical section.**
  - Software Solution: CS conditions (Mutual Exclusion, Progress, Bounded Waiting)
  - `Test_and_Set()`, `Compare_and_swap()`
  - Locks, Semaphores, Condition Variables, Monitors (You need to know how the atomic operations inside each of these are implemented).

- **Understanding of Classic problems in synchronization**
  - Readers-Writers (All versions), Dining Philosopher, Producer Consumer, etc.

- **Spinning vs. blocking. Tradeoff?**
Deadlocks

- **Deadlock. Necessary conditions?**
  - ME, Hold and Wait, No Preemption, Circular Wait

- **Deadlock Prevention?**

- **Deadlock Avoidance**
  - Resource Allocation Graph (Single Resource)
  - Banker’s algorithm.

- I could give you a synchronization code and ask you to look for deadlocks.
Virtual Memory

- Paging. Segmentation. How do they work? Pros and cons?
- Translation Lookaside Buffer (TLB). What is it used for? How does it work?
- Translation from logical (virtual) to physical address. How?
- Page tables. Where are they stored? What can we do to reduce their size?
- Single Level, Multi-level paging, Inverted Page Tables (Pros and Cons)
- Copy-on-write
Virtual Memory

- Page replacement policies: FIFO, LRU, Optimal, 2nd chance, Nth chance.
  - I could give you a sequence of page accesses and ask you for counting the number of page faults, check for belady’s anomaly, etc.

- Understanding Locality.
  - I could give you a program and ask you to optimize it for reducing the page faults

# CPU Scheduling

- **Metrics**: throughput, utilization, waiting-time, turnaround, response time

- **Policies**: FCFS, SJF, RR, Priorities, MLFQ, Lottery scheduling. Pros and cons.

(Source: Wikipedia)

<table>
<thead>
<tr>
<th>Scheduling algorithm</th>
<th>CPU Utilization</th>
<th>Throughput</th>
<th>Response time</th>
</tr>
</thead>
<tbody>
<tr>
<td>First In First Out</td>
<td>Low</td>
<td>Low</td>
<td>Medium</td>
</tr>
<tr>
<td>Shortest Job First</td>
<td>Medium</td>
<td>High</td>
<td>Medium</td>
</tr>
<tr>
<td>Priority based scheduling</td>
<td>Medium</td>
<td>Low</td>
<td>High</td>
</tr>
<tr>
<td>Round-robin scheduling</td>
<td><strong>High</strong></td>
<td>Medium</td>
<td>Low</td>
</tr>
<tr>
<td>Multilevel Queue scheduling</td>
<td><strong>High</strong></td>
<td><strong>High</strong></td>
<td>Medium</td>
</tr>
</tbody>
</table>
Some Examples?

- Third Readers Writers problem
  - No Starvation! How do you design this?