Review for Final Exam

Chapters 1-12

Exam Details

- Final Exam: Comprehensive
  - Closed Book, Closed Notes, One sheet of notes (2 sided)
- Covers Chapters 1 - 12
  - We have skipped parts of each chapters - you are responsible for what we've covered in class
- You will have 3 hours
  - Short Answer/Multiple Choice
  - Longer Problems

System Calls

- OS exposes functionality to applications via **system calls**
  - From a programmer's perspective, they are nothing special…
  - Their implementation is **very** different however…
- *Understand dual-mode execution and interrupt handling*

Processes

- Process v.s Program
  - A Process is a *running* program, with data, stack, heap, etc.
- Process can be in 5 different states
- OS represents each process by a **P**rocess **C**ontrol **B**lock
- *Understand what fork and exec calls do…*

IPC

- Inter-process Communication:
  - Shared memory
  - Message Passing
- Understand pipes and how they can be implemented

Threads

- Threads consist of separate stack, registers, program counter
  - All threads within a process share code and heap
- Kernel v.s. User Threads

- Understand thread creation API calls

Scheduling Review

- A scheduling algorithm defined by two "questions"
  - ***When*** do we put another process onto the CPU?
  - How do we choose ***which*** process gets the CPU?

When?

- When a process terminates
- When a process blocks (I/O)
- When a process yields

Non-Preemptive

- When a new process arrives on ready queue
- After a certain amount of time (quantum)

Preemptive

Which Process?

- First Come, First Serve (FIFO queue)
- Shortest Job First
  - *Shortest Remaining Time First (same algorithm, just with pre-emption*
- Priority - pick highest priority in ready queue
- Round Robin (every process is equal)
  - *In homework, we added priority to RR*

Timing Diagrams

- If asked two write a timing diagram on test (Gantt chart), you will be told *exactly* when to switch processes, and how.

- For example:
  - **When**: terminate, new process, quantum
  - **How**? Shortest First, FCFS, RR (equal), RR (Priority), etc.

Evaluation

- CPU Utilization
- Turnaround time
- Wait Time
- Response Time
- Throughput

You should be able to identify which measures are critical for different situations

Ex.

Batch System
Mainframe/Server
Interactive system

Advanced Scheduling Topics

- Understand the relationship between scheduling implementation within user threads and kernel threads by the OS
- Understand how multiple processors effect scheduling and performance for various thread models
- You do not need to memorize OS priority classes (although you should understand the principles of the Solaris, Windows, and Linux schedulers)

Synchronization

- Understand what a race condition is and be able to identify one
- Understand Critical Section problem and classic solutions:
  - Peterson's Solution, TSL
- Be able to evaluate Critical Section Solutions
  - 4 conditions

Semaphores & Mutex

- Understand how a semaphore is implemented:
  - pseudo-code
  - Counting v.s. Binary Semaphores
- Advantages of a *monitor* implementation

Synchronization

- Deadlocks:
  - Know the 4 requirements of deadlock
    - Invalidate them to prevent.
  - Know the 3 approaches to dealing with them.
    - Understand Avoidance Schemes
      - Resource Allocation Graphs, ~~Banker's Algorithm~~
  - Know the design decisions that are involved in detecting/ correcting deadlocks

Memory Management

- Understand Logical v.s. Physical Address.
  - When are they the same?
    - Compile and Load Time binding
  - When are they different?
    - Run-Time Binding
- Understand how Contiguous Allocation is implemented (in hardware & software)

Memory Management

- Understand the difference between internal and external fragmentation
  - Be able to compute how much fragmentation exists
- Paging: why? how?
  - Why is a TLB useful - what information does it contain?

Paging

- Be able to explain why we would “page the page table”, and how to translate logical addresses with multi-level page systems
- Understand the need and drawbacks of an inverted page table
- ~~Segmentation: Be able to describe why this is used~~
  - ~~Why are hybrid systems more common?~~

Virtual Memory

- Be able to explain the advantages of on-demand paging (and paging in general)
- Define and explain steps required to process a memory request with a page fault.
- Page Replacement Algorithms:
  - FIFO
  - LRU
  - FIFO - second chance.

More Memory Management

- Define and explain thrashing:
  - What causes it, what makes it worse
  - How to *contain* it.
- Be able to discuss the advantages of using working set theory
  - Given a reference string, should be able to define a reasonable working set breakdown
  - Importance of working set size.
- Explain how your code or compiler can minimize paging

Storage Management

- Define the data structures that reside on the disk (and what they are used for):
  - Boot Control Block
  - Volume Control Block
  - File Control Block (inode)
  - DIR structures
- Know what info is stored in memory (OS):
  - Mount Tables
  - Directory Cache
  - Open File Tables

File Allocation

- Be able to compare/contrast the following allocation schemes:
  - Contiguous
  - Linked List
  - File Access Table (FAT)
  - Indexed Allocation (Index tables)
- Tracking Free Space:
  - Bit Vectors (where is this stored?)
  - Linked List

Be able to describe how random access can be supported

Disk Hardware Issues

- Understand the various hardware components involved
- Compare, Contrast, Step through Request Ordering Algorithms
  - FIFO, SSF, SCAN/C-SCAN, LOOK/C-LOOK)
- Be able to describe bad block recovery techniques (Error Correction Codes)

Distributed Systems

- Be able to define, compare, contrast the 3 classes of multiprocessor systems:
  - Multiprocessor, Shared Memory
  - Multi-Computer, Message Passing
  - Distributed Systems

Good Luck!

- Good luck on all of your exams!