# CSC3350 Final Exam Review Sheet

Introduction

1. Overview
   1. Operating Systems Programming
      1. Motivations
      2. Theory and Practice
   2. Environments:
      1. Win -> WSL 2
      2. Mac -> Terminal (POSIX based)
   3. Shell
      1. What is a shell?
      2. Different shells: bash, dash, csh, ksh, zsh
   4. Programming Language
      1. Why the C programming language?
2. Dev Environment: Linux
   1. Why Linux?

Virtualizing the CPU

1. Processes
   1. Why do we want to virtualize the CPU?
   2. What is a process? Why do we have them?
   3. What makes up a process? What is the essential process data?
   4. Process Creation
      1. How does it work?
      2. When does it happen?
   5. Process States and Process State Transitions
      1. What are they?
      2. How do they work?
   6. PCB: Process Control Block
      1. What does it contain?
      2. Where is it kept?
      3. When and how is it updated?
   7. PT: Process Table
      1. What does it contain?
      2. Where is it kept?
      3. When and how is it updated?
2. Process API
   1. System Call:
      1. What is it?
      2. How do they work?
   2. POSIX
   3. fork(), exec(), wait(), exit()
   4. system()
      1. How is system() different from fork()/exec()?
   5. Able to read, write, and debug code using: fork(), exec(), wait, AND system()
   6. Waiting for children
      1. Purpose
      2. What happens when the parent doesn’t wait for a child process?
   7. What happens during exec()?
   8. Why are fork(), exec(), and system() each weird functions?
   9. How does the shell work?
3. Limited Direct Execution
   1. Direct execution -vs- Limited Direct Execution
   2. How does Direct Execution work?
   3. PE: Portable Executable (.exe) file layout
      1. Header, .text, .bss, .rdata, .debug sections
      2. Where are the code, heap, and stack?
   4. Limited Direct Execution
      1. Motivation: Problems with Direct Execution
      2. How does LDE solve them?
      3. Privilege Levels
         1. How do they work?
         2. What is the difference between kernel mode and user mode?
         3. How are they used?
      4. The trap Instruction
         1. What does it do?
         2. Why is it special?
   5. Process Switching
      1. Separation of Mechanism and Policy
         1. Mechanism: Dispatcher
         2. Policy: Scheduler
   6. Dispatcher
      1. Basic concept
      2. How does the Dispatcher gain control?
         1. Special hardware: What is the special hardware?
      3. How does the process switch happen?
         1. What information needs to be saved?
            1. Where does the information come from?
            2. Where is it saved?
            3. How and when is it restored?
         2. Limited Direct Execution Protocol
            1. Be able to describe it, list the steps, and trace or debug a list of steps
4. Scheduling
   1. Recap Dispatcher and Scheduler
   2. What metrics are we trying to optimize?
      1. Be able to list all optimization metric
      2. And be able to compute them from an example
   3. Scheduling Policies
      1. FIFO: First In First Out
      2. SJF Shortest Job First
      3. STCF: Shortest Time-to-Completion First
         1. Aka: PSJF: Preemptive Shortest Job First
      4. RR: Round Robin
      5. Be able to describe and compare/contrast them
      6. Be able to list the advantages and shortcomings of each
   4. New Metric: Response Time
      1. Be able to compute it from an example
      2. How to Optimize Response Time
         1. RR: Round Robin
   5. How does doing IO change things?
      1. Blocking -vs- Non-Blocking system calls
      2. Incorporating IO into the scheduling policy
5. Multi-Level Feedback Queues
   1. Define MLFQ and give an example of how it works
   2. Be able to list and define the 5 MLFQ Rules
   3. The Queues: How are the process queues organized and used?
   4. Priorities
      1. How are they assigned?
      2. How do they change (based on the 5 rules)?
   5. Scheduling Recap:
      1. The 6 types of CPU Scheduling Algorithms
      2. CPU Scheduling: Preemptive -vs- Non-Preemptive
      3. Scheduling Criteria
         1. Maximize
            1. CPU Utilization
            2. Throughput
         2. Minimize
            1. Turnaround Time
            2. Waiting Time
            3. Response Time

Virtualizing Memory

1. Virtualizing Memory
   1. Why do we want to virtualize the memory?
   2. History and Evolution
      1. Uniprogramming -vs- Multiprogramming
   3. The 4 Goals of Memory Virtualization
   4. Abstracting the Address Space
      1. Code, Heap, and Stack
   5. Memory API
      1. malloc() -vs- new()
      2. free() -vs- delete()
   6. Stack Organization and Fragmentation
   7. Heap Organization and Fragmentation
2. Address Translation
   1. Motivation - Problems with Time-Sharing Memory
   2. Need for Address Translation
      1. All processes think they have access to all the virtual memory
      2. Application developers don’t have to think about physical memory, only their private virtual memory space. Be able to describe and explain why.
   3. How Address Translation Works
      1. Be able to describe how to transform a virtual address into a physical address
      2. Be able to trace/debug an example
   4. Base and Bounds
      1. What are they?
      2. How do they work?
   5. Static Relocation
      1. What is it?
      2. How does it work?
      3. Disadvantages
   6. Dynamic Relocation
      1. What is it?
      2. How does it work?
      3. Why is special hardware needed?
   7. MMU: Memory Management Unit
      1. What is it?
      2. Why is it needed?
      3. How does it work?
   8. Multiple Relocated Processes
      1. How does it work?
   9. Address Translation
      1. Dynamic Translation: Base and Bounds
      2. Be able to describe how it works and trace/debug an example
   10. Segmentation
       1. Motivation – Why create and manage segments of memory?
       2. Generalized Base and Bounds per segment
       3. Segment Protection
          1. Read-Execute -vs- Read-Write
          2. Why have different protections?
          3. What is the protection of the code, heap, and stack segments?
       4. Segment Grow Direction
          1. Grow Positive or Grow Negative
          2. Why have different growth directions?
       5. Segmentation Registers
          1. How are Segment Protection and Growth directions encoded in Segmentation Registers?
   11. Segmentation Violation: SIGSEGV
       1. What is it?
       2. What causes it?
       3. What happens when a SIGSEGV occurs?
3. Paging
   1. Motivation
      1. Running out of physical memory
      2. Sharing memory across multiple processes
      3. Advantages of Paging
   2. Simple Paging
      1. Be able to describe how paging works and sketch a paging example
      2. Be able to trace or debug a paging example drawing
   3. Address Translation
      1. VPN and Offset
      2. Mapping a virtual address to a physical address
      3. Be able to describe how Address Translation works
      4. Be able to sketch a Paging Address Translation example
      5. Be able to trace or debug a paging example
   4. Page Tables
      1. What are they?
      2. What do they contain?
      3. Where are they stored?
      4. How are they updated?
   5. PTE: Page Table Entry
      1. What are the elements of a PTE?
      2. How are they stored?
      3. When and how are they updated?
   6. Paging is slow
      1. How do we speed it up?
      2. Why is special hardware required?
      3. What hardware do we use to speed up paging?
4. TLB
   1. Paging and “Modern” Virtual Memory
      1. Page Table
         1. VPN (Virtual Page Number) and
         2. PFN (Page Frame Number, aka Physical Page)
   2. Translating Addresses
      1. Translating a Virtual Address [VPN, Page Offset] to a Physical Address
   3. The TLB - Translation Lookaside Buffer
      1. What is it?
      2. How is it implemented?
      3. Where is it located?
      4. Define and describe how a CAM (Content Addressable Memory) cache works.
   4. The MMU: Memory Management Unit
      1. What is it?
      2. When is it accessed?
   5. Be able to describe how TLB lookup works
      1. Hits and Misses
      2. What happens in each?
      3. What is the relative cost of Hits -vs- Misses?
   6. The makeup of a TLB entry: VPN, PFN, and the Other Bits
   7. Be able to describe the TLB Algorithm with pseudocode, including Hit and Miss cases
   8. Be able to count Hits and Misses when accessing memory, e.g., an array
   9. The TLB and Context Switching
      1. What is ASID?
      2. Where is it stored and updated?
      3. What is its role in context switching?
   10. Sharing a page of physical memory across multiple processes
       1. Why is it desired?
       2. Why is it dangerous?
       3. How does it work?
   11. Message Passing -vs- Shared Memory
       1. How does each work?
       2. Producer/Consumer
          1. What is it?
          2. How does it work (basic level)?
   12. Locality Types:
       1. Temporal and Spatial
       2. Be able to define and provide examples
   13. Page Replacement Policies
       1. List and describe tradeoffs of the 3 primary types
       2. Belady’s Anomaly
       3. Be able to count Hits and Misses of different Page Replacement Policies in an example

Concurrency

1. Concurrency - Threads
   1. Processes -vs- threads
      1. What is the difference?
      2. What do they share, not share?
      3. Why Threads?
         1. Advances in hardware have provided multi-CPUs and multiple cores on a CPU, allowing threads to execute in parallel simultaneously
         2. How does the schedule manage threads?
   2. PCB -vs- TCB
      1. Compare and contrast
   3. Thread API
      1. POSIX
         1. pthread\_create()
         2. pthread\_join()
         3. Note and understand the difference with the OSTEP textbook versions that begin with an upper-case “P”
      2. Coding examples of creating and joining thread
   4. What about shared data?
      1. Critical code sections
      2. Race Condition
         1. What is it?
         2. How do we avoid it?
2. Locks
   1. Locks: The Basic Idea
      1. Critical Section of Code
      2. Mutex
         1. What does it mean?
         2. How is it used?
   2. Lock Semantics
      1. Lock()
   3. pthread Locks – mutex
      1. How do you define a mutex lock?
      2. What are the pthread calls to use a mutex lock?
   4. Basic Lock Criteria
      1. List 3 of them
   5. Early Lock Implementations
      1. Why is additional hardware needed?
      2. Two problems
   6. Locks: Coding and Evaluation
      1. Test And Set (Atomic Exchange)
         1. Be able to code TestAndSet()
      2. Spin Lock
         1. Be able to code a Simple Spin Lock using test-and-set
         2. Spin Lock Evaluation
            1. How do test-and-set spin-locks satisfy the 3 lock criteria
      3. Compare and Swap
         1. Be able to code A Simple Spin Lock using compare-and-set
         2. Compare-and-Set Evaluation
            1. How do compare-and-set spin locks satisfy the 3 lock criteria
3. Locks and Lock Based Data Structures
   1. Fetch-and-Add
      1. What is it?
      2. Be able to code it using compare-and-swap
      3. What is the problem with this approach?
      4. What is the solution?
   2. Ticket Locks
      1. Be able to code a ticket lock.
      2. How does one avoid so much spinning?
         1. Simple approach
            1. Be able to explain and code it
            2. What are the drawbacks of the simple approach?
         2. Improved Approach – Sleeping instead of spinning
            1. Be able to explain and code it
            2. What data structure is used to improve things? How does it work?
            3. Implementations

Solaris

futex

* + - * 1. What are the drawbacks of this improved approach?

1. Condition Variables
   1. Condition Variable introduction
      1. What are the other types of thread synchronization?
      2. What is the data structure used to build a CV?
      3. Be able to pseudocode a CV example:
         1. Spin-based approach
   2. Waiting -vs- Signaling: key concepts and definitions
   3. Parent waiting for a child
      1. Be able to code an example in POSIX
      2. What is the importance of the state variable?
      3. Be able to point out the drawbacks of different coding implementations
   4. Producer/Consumer (Single buffer)
      1. Be able to pseudocode and POSIX code key Producer/Consumer examples
         1. Two condition variables
         2. Threads (version 3)
         3. What if we have more than one producer or consumer?
      2. Thread Tracing
         1. Be able to trace the code of this particular thread implementation
         2. Be able to analyze and identify issues in this thread trace
      3. What is the drawback of this approach?
2. Semaphores – Basics
   1. What is a semaphore?
      1. Be able to define and explain
      2. Be able to write POSIX code to define and use a semaphore
         1. sem\_init()
         2. sem\_wait()
         3. sem\_post()
   2. Binary Semaphores (locks)
      1. Be able to define and explain
      2. Be able to write POSIX code to define and use a binary semaphore
      3. Thread tracing
         1. Be able to trace a single thread using a semaphore
         2. Be able to trace two threads using a semaphore
   3. Semaphores as Condition Variable
      1. Be able to define and explain
      2. Be able to write POSIX code to define and use a semaphore
      3. Thread tracing
         1. Be able to trace two threads using a semaphore
            1. Parent waiting for a child (case 1)
            2. Parent waiting for a child (case 1)
   4. Producer/Consumer (Bounded buffer)
      1. Be able to define and explain
      2. Be able to write POSIX code to define and use a semaphore
      3. Be able to describe issues and drawbacks
      4. Provide a solution
         1. Why does the initial attempt to add mutex solve the problem?
         2. What issues still existed/
         3. How does the final solution solve the problems
            1. Be able to write POSIX code to define and use a semaphore
3. Semaphores – Advanced
   1. Reader-Writer locks
      1. Be able to define and explain
      2. Be able to write POSIX code to define and use a reader-writer lock
      3. What problems/drawbacks do reader-writer locks have?
   2. The Dining Philosophers
      1. Be able to define and explain The Dining Philosophers problem
      2. Be able to write POSIX code The Dining Philosopher problem using semaphores
   3. Zemaphores
      1. Be able to define and explain
      2. Be able to write POSIX code to define and use zemaphores
4. Semaphores - Exercises
   1. Basic semphore pseudocode
   2. Rendezvous
      1. Be able to define and explain
      2. Be able to write POSIX code to implement Rendezvous
      3. Be able to describe issues and drawbacks
   3. Mutual Exclusion
      1. Be able to define and explain
      2. Be able to write POSIX code to implement Mutual Exclusion
      3. Be able to describe issues and drawbacks
   4. Compare and Contrast Rendezvous and Mutual Exclusion
      1. How are they similar?
      2. How are they different?

Persistence

1. Input and Output
   1. I/O Devices
      1. What are they, and why are they needed?
      2. What are the key issues
   2. Architecture and Buses
      1. What is a bus?
      2. How is it used with I/O? Devices?
      3. Be able to define, describe, and explain I/O Architecture
   3. The Hardware Interface of a Canonical Device
      1. What are the key hardware registers involved?
      2. Be able to pseudocode and code interacting with a canonical hardware device
   4. The 3 Basic I/O Techniques
      1. What are the 3 Basic I/O Techniques?
      2. PIO (Programmed I/O)
         1. Be able to define, describe, and explain
         2. Be able to trace/sketch how it works
         3. Is special hardware required? Explain and describe.
      3. Interrupt Drive I/O
         1. Be able to define, describe, and explain
         2. Be able to trace/sketch how it works
         3. Is special hardware required? Explain and describe.
      4. DMA (Direct Memory Access)
         1. Be able to define, describe, and explain
         2. Be able to trace/sketch how it works
         3. Is special hardware required? Explain and describe.
      5. Compare and contrast: PIO, IIO, and DMA
         1. Rank them from best to worst. What criteria should be used for the ranking?
         2. Polling -vs- Interrupts
            1. Where does this come to play?
            2. Which is better? Provide examples.
2. Files and File Systems
   1. Device Communication
      1. Be able to list, describe, and explain the File System Stack
      2. How does the OS communicate with devices?
         1. Be able to give x86 examples:
            1. PIO
            2. DMA
   2. Hard Disk drive (HDD) Internals
      1. HDD (Hard Disk Drives)
         1. Platters, tracks, sectors, rotational speed
         2. Be able to draw the geometry of an HHD
      2. Accessing a sector
         1. What is involved?
         2. Be able to draw and describe
   3. Solid State Drive (SSD) Internals
      1. Address space, sectors, access method
      2. Be able to draw the geometry of an HHD
   4. Compare and contrast HDD and SSD
      1. The 7 comparison categories
   5. Time required for an I/O Operation
      1. Be able to describe and calculate the time of an HDD read/write
   6. Disk Scheduling
      1. The drawbacks of FIFO scheduling
      2. Disk Scheduling Algorithms
         1. List and describe them
         2. Compare and contrast
         3. Which is the best algorithm? Why?
   7. File Abstractions
      1. File -vs- Directory
         1. Define and compare & contrast
      2. Directory Tree
   8. Operations on Files
      1. Be able to define and explain the key operations
      2. Be able to write POSIX code to operate on files
      3. Be able to describe issues and drawbacks
      4. How are directories similar/different from files?
   9. Hard Links -vs- Soft Links (Symbolic Links)
      1. Be able to define and explain the key similarities and differences
      2. Be able to write Linux code for both
      3. Be able to describe issues and drawbacks
   10. Filesystems
       1. Filesystem Implementation
          1. Be able to define and explain
          2. Be able to write Linux commands to manipulate the filesystem
          3. Be able to describe key issues and drawbacks
       2. Mounting a Filesystem
          1. Be able to define and explain the key similarities and differences
          2. Be able to write Linux commands to mount, unmount, and list mounted filesystems
          3. Be able to describe issues and drawbacks
       3. File Organization
          1. Be able to define and explain the key elements
          2. Be able to draw what it looks like
          3. Be able to describe key issues and drawbacks
       4. Inodes Basics
          1. Be able to define and explain the key elements
          2. Be able to draw what it looks like
          3. Be able to describe key issues and drawbacks
       5. File Organization: The Inode
          1. Be able to define and explain the key elements
          2. Be able to draw what it looks like
          3. Be able to describe key issues and drawbacks
       6. Directory Structure: The Inode
          1. Be able to define and explain the key elements
          2. Be able to draw what it looks like
          3. Be able to describe key issues and drawbacks
       7. Free space management
          1. Be able to define and explain the key elements
          2. Be able to draw what it looks like
          3. Be able to describe key issues and drawbacks