CSCI330: Final Prep Document

Rules and Logistics

- 2 hour time limit, on Sakai
- Open lecture slides, your notes, textbook
    - Closed [other] internet
    - Don't download Wikipedia or other web pages into your notes
- Write your answers in Word and then copy to Sakai to avoid issues with the site timing out

Everything from the first exam

- necessarily cumulative BUT focus is on the material since the last exam
- bring together all of the ideas from the course (well, all of the most important ones)

Thread Synchronization

- Goals
    - Safety, liveness, mutual exclusion, …
- Problems: race conditions, critical sections, deadlock
- Mechanisms
    - Locks, mutexes, condition variables, semaphores, monitors
        - Uses, APIs
    - Atomic instructions
- Synchronization problems (producer-consumer, taking turns, dining philosophers), solutions

File Systems

- Goals
- Abstractions
- Files vs File Systems
- Directories
- Data Structures
    - Metadata, inodes, superblock, …
- ~~Fragmentation~~
- Functionality/API

Storage Management

- Disk management, Storage
- Disk Scheduling
    - Policies, tradeoffs
- RAID

Memory Management

- Goals
- Mechanisms
    - VM, Paging, Segmentation, Swapping
    - Purposes, Tradeoffs

- Hardware support
    - MMU, TLB
- Policies (~~allocation~~, ~~free space~~, replacement, demand paging, ~~prefetching~~)
    - Purposes, Tradeoffs
- Challenges
    - Fragmentation, ~~Thrashing~~
- Locality – ~~spatial,~~ temporal
- Optimizations

OS Project

- Building: make, Makefile, compiling, assembling
- Source code: C, data structures
- Version Control

What I expect from you on exam

- To know the material well and just need your notes as backup/clarification
- To be able to synthesize the material. We spent multiple class periods plus projects/assignments on various topics. Bring it all together.
    - What are the common issues/challenges? What are common solutions and their tradeoffs? Why/when are certain tradeoffs made? What caused the tradeoff to be deemed okay?
- To be able to state many of these ideas relatively succinctly, hitting on the salient points.
- To be able to analyze problems, suggesting solutions, and articulating tradeoffs/limitations
- To be able to read code written in C, Bash, or a Makefile and describe what it does (e.g., describing its output, identifying where there is a problem, …)
- To be able to analyze synchronization problems, identify the issues, and write pseudocode to solve the synchronization problem

Suggestions on how to prepare

- Review your notes and my slides; the textbook may be helpful for clarification—especially deeper discussion and details that I can't fit on slides
    - What are common threads/issues/challenges/solutions and their tradeoffs?
- Review code examples, run them, explain their output
- Review the projects and assignments
    - What were the objectives?
    - Did you understand their results?