CSCI 315 Operating Systems Design

Final Exam Study Guide

- This document doesn't mean to *tell you what will be on the final exam*. Instead, it means to guide your learning and preparation for this summative assessment.
- The final exam is designed to take only 60 minutes within the period allocated for the final exam.
- The final exam is open book and open notes. You are strongly encouraged to create a one sheet summary of information to save time in searching through other material.
- The final exam emphasizes the latter part of the material (since Midterm 2), but it is cumulative in the sense that we cannot sidestep important concepts that were covered earlier in the class (including but not limited to: processes, threads, concurrency, synchronization, dispatching, and scheduling).
- The exam will include concepts and operational aspects that were featured in lab assignments (for instance: problems may ask you to demonstrate that you can apply system calls, library functions, and features of the C programming language). Unix manual pages for the system calls and library functions will be provided with exam questions that cover them.
- Review the textbook chapter reading guides and ensure that you can answer the questions they pose.
- Review the problems featured in activities so as to make sure you can solve them without much trouble.

---

1. Explain, compare, and contrast the following OS design concepts:
    - disk partition
    - disk block
    - file
    - file control block
    - file system
    - file system consistency check
    - directory
    - file attributes
    - contiguous allocation of file space
    - linked allocation of file space
    - indexed allocation of file space
    - Unix “combined scheme” or inode
    - disk management of free space management
        - bitmap
        - linked list
        - grouping
        - counting
    - buffer
    - symbolic link
    - hard link
    - software cache

2. File system organization: compare the pros and cons of single-level, two-level, tree, acyclic-graph, and general graph structured systems. Identify use cases where each of these structures is desirable.

3. In the context of Unix file systems, what do the file attributes for *owner*, *group*, and *other* express? How do they relate to protection? How do you manipulate these bits using the chmod command?

4. How does Unix convert the numerical identifiers for *owner* and *group* stored in an FCB (inode) to human readable strings?

5. Why doesn’t the publicly readable file **`/etc/passwd`** show user passwords?

6. Why is the file **`/etc/shadow`** made unreadable to all users except for the system administrator?

7. What problems can occur when links to directories are allowed in a directory structure?

8. What is the difference between *hard links* and *symbolic (soft) links*? What use cases do they serve?

9. Compare the methods for space allocation for files. What are the advantages and drawbacks of each?

10. Compare the following methods for organizing the disk blocks allocated to files: contiguous, linked, FAT, indexed, and Unix inode. What is the overhead associated with each method? Does this overhead increase with file size?

11. Compare the following methods for organizing the free blocks in a disk: bitmap, linked list, counting, and grouping.

12. Consider the data that is stored in a disk to implement free space management using bitmap, linked list, counting, and grouping. Explain how much data space is required for each method and whether that space can be considered *overhead* (in the sense of taking up space that might otherwise be used for data storage).

13. Does the Unix inode impose limits on file size? What overhead does the inode require for small files? How does this overheard grow with the size of files?