

CS 241 Final Exam Study Guide

The class final examination will be held in **from 8:00 to 11:00 a.m. on Tuesday, December 13th**. Room locations will be announced soon. The exam will begin and end promptly. Please arrive before 8:00 to allow everyone to settle into their seats before the test begins. No extensions will be granted to those who are late, nor will any non-emergency excuse for absence be accepted after Monday, December 5th.

You may not consult any materials during the exam: no textbooks, no crib sheets, no calculator, etc.

The final will contain a set of around 30-35 multiple choice questions and 7-10 long answer questions. About half the total points on the exam will be for each type of problem. Each long answer may consist of multiple parts and will carry approximately equal weight overall, but may break down unevenly amongst the parts. These questions may ask you to reason about or write some code or to determine the behavior of an algorithm. On the long answer questions on the midterm, you must show all work and reasoning, writing both work and solution legibly, and should box all answers. If the course staff cannot read a solution, no credit will be given. ***All questions on the midterm will all be based on the questions on this study guide, the midterm study guide, in Homework 1, or based on MPs 1-8.*** The breakdown will be about 30% from the first half of the semester and 70% from the second half.

I. Processes and Deadlock

1. How would the implementation of a web server using threads differ from one using processes?
2. What can happen if synchronization in a multiple-threaded program is not programmed carefully?
3. Why does the operating system use a resource allocation graph?
4. What are the conditions of a deadlock? How could you guarantee that each one of these conditions can be prevented?
5. What does `waitpid()` do?
6. What are the approaches for solving deadlock?
7. What is the difference between Deadlock Prevention, Deadlock Detection & Recovery, and Deadlock Avoidance? What deadlock handling mechanism would you use?
8. What are the components of a resource allocation graph?
9. What problem does the Banker's Algorithm solve? Given a set of processes how would you use the Banker's Algorithm?
10. What is a safe state and how can you determine if a system is in a safe state?

II. IPC

1. What is the difference between a FIFO and a pipe?
2. How would you redirect standard out to a file?
3. What is the difference between a pipe and an ordinary file on disk?
4. What happens when two processes read and write to a memory mapped file?
5. Explain how two processes can share memory using `shmem`.
6. Explain how a process can set which signals are caught or ignored using a signal set.
7. How can one process send a signal to another?
8. Describe the purpose of a POSIX signal.

9. Some signals cannot be caught or ignored. Which signals are they and why shouldn't they be allowed to be caught?
10. What does "`kill -<parameter> pid`" do?
11. How is the function `sigwait()` used?
12. How does the function `alarm()` work?

III. Networking

1. When do you use the `close()` system call with sockets?
2. Discuss how a multithreaded web server running on a single processor system could be optimized using the process scheduling methods discussed in class? Which do you recommend?
3. How does `select()` work? What problem does it solve?
4. Describe the Posix accept function.
5. How does HTTP work?
6. Describe the services provided by TCP?
7. How does TCP connection establishment work?
8. Describe the services provided by UDP?
9. Explain the difference between a regular and a connected UDP socket.
10. How does sockets programming support the client-server model?
11. Which is better, UDP or TCP? Which one would you use?

IV. Memory

1. What is the difference between physical and virtual memory?
2. What are the different memory allocation selection algorithms and what are the advantages of each?
3. How are virtual addresses translated to physical addresses in multi-level page tables?
4. How do page size and the number of levels of page tables affect the number of entries in a page table?
5. What is the difference between internal and external fragmentation?
6. What are the different page replacement policies and the advantages of each?
7. Describe how the buddy system works and the run time for different operations.
8. What is thrashing? When does it occur?
9. What causes a SEGFAULT and what happens when one occurs?
10. When is a process swapped out to disk?
11. What is the difference between the MMU and the TLB? Describe the function of each one.
12. Why is the TLB fully-associative?
13. Name three benefits of virtual memory (as opposed to allowing programs to directly access physical memory).
14. Name one advantage of segmentation over paging, and one advantage of paging over segmentation.
15. How is a page table similar to an inode? What is the difference between these structures?
16. Assuming a 32-bit address space and 4 KB pages, what is the virtual page # and offset for virtual address 0xd34f6a5?
17. Give an example of a page fault that is an error, and an example of a page fault that is not an error.

18. Assume LRU eviction. Describe what happens when the application accesses virtual memory pages in this sequence: 3 (shown), 4,5,4,1,6,9,3,9,8,4,8,8,2.
19. How many page faults occur?
20. Why are pages set to read-only in the copy-on-write technique?
21. Suppose we have a 64-bit address space and 16 KB pages. How big is the page table of a single process? What is the problem here? How would multi-level page tables help solve this problem?
22. Which scheme is better OPT or LRU? Why?
23. Why does LRU not suffer from Belady's anomaly?
24. How does the virtual memory subsystem know the exact location where a particular page is stored on disk, if it is swapped out of memory?
25. How is the working set computed? How is the notion of a working set useful for managing memory of processes?
26. The free list schemes described in class are singly-dimensional. How would you design free lists for a memory indexed by a two-dimensional address? Include a description of what metadata you would store, and how you would use that metadata to perform coalescing in a manner that does not lead to false fragmentation.
27. Compare and contrast (give one benefit and one disadvantage) for: implicit, explicit, segregated, and buddy free lists.

V. File systems and I/O

1. Given a description of the block size and i-node structure, what is the maximum size of a file?
2. How many i-node operations are required to fetch a file at /path/to/file?
3. What information is stored in an i-node? What information isn't?
4. What data structure best describes an i-node?
5. What are the advantages and disadvantages of an i-node based filing system?
6. Given the description of an i-node filing system, how many i-node accesses are required to read the entire contents of a file of a given size? How many blocks does this file consume on disk?
7. What is the advantage of a soft link over a hard link?
8. What is I/O polling? What are advantages and disadvantages?
9. Describe disk I/O access using DMA.
10. How are file descriptors shared between threads in a single process? How are they shared between after a process executes a fork()?
11. Describe an i-node filing system.
12. When the size of a block changes in an i-node based filing system, how does this change the maximum size of a file?
13. How does polling and interrupt driven I/O differ? What are the advantages and disadvantages of each.
14. How does the page-out process work?
15. Understand how hard-links result in different file names affecting the same i-node.
16. If an i-node based filing system has a certain number of full direct and single-indirect blocks, how large is the file?
17. Describe the relationship between processes, children and their file descriptors.