CS 241 Fall 2012

Midterm Exam Study Guide

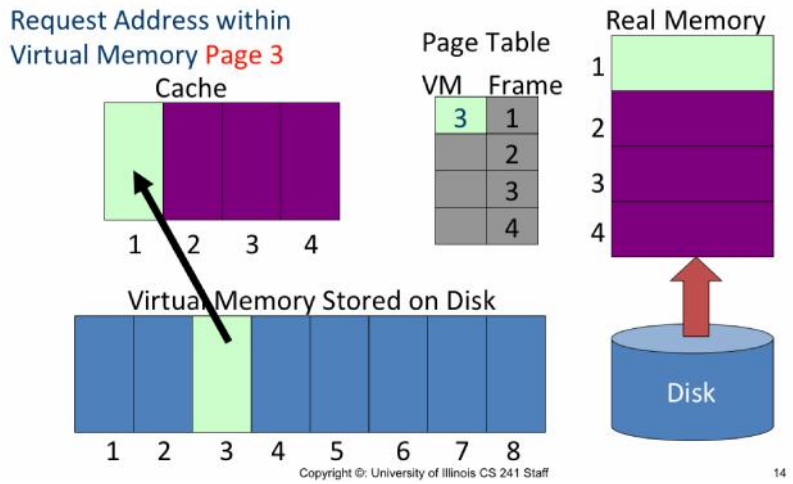
C Programming

1. What is POSIX?
2. Explain the difference between a library function and a system call.
3. Give an example of a POSIX system call used in CS241 lectures or reading.
4. How does pointer arithmetic work?
5. What is the * operator? What does it do? What is the & operator? What does it do?
6. How do you define a function pointer?
7. What is a String? What is NULL?
8. What is the difference between strlen and sizeof?
9. What's the difference between a stack and a heap variable? What about global and static variables?
10. When is a stack full?
11. How does malloc and free work?
12. What's the difference between char c[80] and char *c? …what about when they're used in sizeof()?
13. What is the difference between a string and a string literal?
14. How do strcpy, strcat, and strncat work?

Memory

15. What is the difference between physical and virtual memory?
16. What are the different memory allocation selection algorithms and what are the advantages of each?
17. How are virtual addresses translated to physical addresses in multi-level page tables? Given a virtual address and the physical page number, can you write code to translate the virtual address to a physical address?
18. How do page size and the number of levels of page tables affect the number of entries in a page table?
19. What is the difference between internal and external fragmentation?
20. What are the different page replacement policies and the advantages of each?
21. Describe how the buddy system works and the run time for different operations.
22. What is thrashing? When does it occur?
23. What causes a SEGFAULT and what happens when one occurs?
24. When is a process swapped out to disk?
25. What is the difference between the MMU and the TLB? Describe the function of each one.

26. Name three benefits of virtual memory (as opposed to allowing programs to directly access physical memory).
27. Name one advantage of segmentation over paging, and one advantage of paging over segmentation.
28. Assuming a 32-bit address space and 4 KB pages, what is the virtual page # and offset for virtual address 0xd34f6a5?
29. Give an example of a page fault that is an error, and an example of a page fault that is not an error.
30. Given the following diagram:



31. 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.
32. How many page faults occur?
33. Why are pages set to read-only in the copy-on-write technique?
34. 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?
35. Which scheme is better: OPT or LRU? Why?
36. Why does LRU not suffer from Belady's anomaly?
37. 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?
38. What is the *working set* of a process? How is the notion of a working set useful for managing memory of processes?
39. Compare and contrast (give one benefit and one disadvantage) for: implicit, explicit, segregated, and buddy free lists.

Threads and Processes

40. Which resources are shared between threads of the same process? Which are not shared?
41. Write a simple program using pthread_create(). What is the possible output of your code?
42. X is a global variable and initially X = 0. What are the possible values for X after two threads both try to increment X?

43. What happens when a thread calls exit()?
44. What happens to a process's resources when it terminates normally?
45. Describe what happens when a process calls fork().
46. Under what conditions would a process exit normally?
47. Explain the actions needed to perform a process context switch.
48. Explain the actions needed to perform a thread context switch.
49. What are the advantages and disadvantages of kernel-level threads over user-level threads?
50. Compare the use of fork() to the use of pthread_create().
51. In a multiprocessor system, what conditions will cause threads within a process to block?
52. Explain how a process can become orphaned and what the OS does with it. How about zombies?
53. Describe how to use the POSIX call wait().
54. Explain what happens when a process calls exec().
55. Why aren't some functions thread safe?
56. What are the maximum number of threads that can be run concurrently? How is this number determined?
57. If a process spawns a number of threads, in what order will these threads run?
58. Explain how to use pthread_detach() and pthread_join() and why these are used.
59. Explain how a shell process can execute a different program.
60. Explain how one process can wait on the return value of another process.
61. Describe the transitions between running, ready and blocked in the 5 state model.

Scheduling

62. Which policies have the possibility of resulting in the starvation of processes?
63. Which scheduling algorithm results the smallest average wait time?
64. What scheduling algorithm has the longest average response time?
65. Define turnaround time, waiting time and response time in the context of scheduling algorithms?
66. Why do processes need to be scheduled?
67. What is starvation?
68. What is response time? What other metrics do we use to evaluate scheduling algorithms?
69. Which scheduling algorithm minimizes average initial response time? Waiting time? Turnaround time?
70. Why are SJF and Preemptive SJF hard to implement in real systems?
71. What does it mean to preempt a process?
72. What does it mean for a scheduling algorithm to be preemptive?
73. Describe the Round-Robin scheduling algorithm. Explain the performance advantages and disadvantages.
74. Describe the First Come First Serve (FCFS) scheduling algorithm. Explain the performance advantages and disadvantages.
75. Describe the Pre-emptive and Non-preemptive SJF scheduling algorithms. Explain the performance advantages and disadvantages.

76. Describe the Preemptive Priority-based scheduling algorithm. Explain the performance advantages and disadvantages.
77. How does the length of the time quantum affect Round-Robin scheduling?
78. Define fairness in terms of scheduling algorithms. What are the fairness properties of each of the scheduling disciplines discussed in class?
79. Which scheduling algorithms guarantee progress?
80. A process was switched from running to ready state. Describe the characteristics of the scheduling algorithm being used.
81. Which properties of scheduling algorithms affect the performance of interactive systems?

Semaphores / Mutexes

82. When do you have to use a semaphore or mutex?
83. What is mutual exclusion?
84. How can use use a mutex lock to ensure that concurrent code operates correctly?