CS 241 (Summer 2012) Midterm Study Guide

C Programming

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

Memory

- What is the difference between physical and virtual memory?
- What are the different memory allocation selection algorithms and what are the advantages of each?
- 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?
- How do page size and the number of levels of page tables affect the number of entries in a page table?
- What are the different page replacement policies and the advantages of each?
- Describe how the buddy system works and the run time for different operations.
- What is thrashing? When does it occur?
- What causes a SEGFAULT and what happens when one occurs?
- When is a process swapped out to disk?
- Name three benefits of virtual memory (as opposed to allowing programs to directly access physical memory).
- Name one advantage of segmentation over paging, and one advantage of paging over segmentation.
- Assuming a 32-bit address space and 4 KB pages, what is the virtual page # and offset for virtual address 0xd34f6a5?
- Give an example of a page fault that is an error, and an example of a page fault that is not an error.
- Given four physical blogs of ram and the sequence of virtual page accesses, 3,4,5,4,1,6,9,3,9,8,4,8,8,2, how many page faults occur in various schemes?
- Why are pages set to read-only in the copy-on-write technique?
- 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?
- Which scheme is better: OPT or LRU? Why?
- 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?
- Compare and contrast (give one benefit and one disadvantage) for: implicit, explicit, segregated, and buddy free lists.

Threads and Processes

- Which resources are shared between threads of the same process? Which are not shared?
- Write a simple program using pthread_create(). What is the possible output of your code?

CS 241 (Summer 2012) Midterm Study Guide

- X is a global variable and initially X = 0. What are the possible values for X after two threads both try to increment X?
- What happens when a thread calls exit()?
- What happens to a process's resources when it terminates normally?
- Describe what happens when a process calls fork().
- Under what conditions would a process exit normally?
- Explain the actions needed to perform a process context switch.
- Explain the actions needed to perform a thread context switch.
- Compare the use of fork() to the use of pthread_create().
- In a multiprocessor system, what conditions will cause threads within a process to block?
- Explain how a process can become orphaned and what the OS does with it. How about zombies?
- Describe how to use the POSIX call wait().
- Explain what happens when a process calls exec().
- What are the maximum number of threads that can be run concurrently? How is this number determined?
- If a process spawns a number of threads, in what order will these threads run?
- Explain how to use pthread_detach() and pthread_join() and why these are used.
- Explain how a shell process can execute a different program.
- Explain how one process can wait on the return value of another process.
- Describe the transitions between running, ready and blocked in the 5 state model.

Scheduling

- Which policies have the possibility of resulting in the starvation of processes?
- Which scheduling algorithm results the smallest average wait time?
- What scheduling algorithm has the longest average response time?
- Define turnaround time, waiting time and response time in the context of scheduling algorithms?
- Why do processes need to be scheduled?
- What is starvation?
- What is response time? What other metrics do we use to evaluate scheduling algorithms?
- Which scheduling algorithm minimizes average initial response time? Waiting time? Turnaround time?
- Why are SJF and Preemptive SJF hard to implement in real systems?
- What does it mean to preempt a process?
- What does it mean for a scheduling algorithm to be preemptive?
- Describe the Round-Robin scheduling algorithm. Explain the performance advantages and disadvantages.
- Describe the First Come First Serve (FCFS) scheduling algorithm. Explain the performance advantages and disadvantages.
- Describe the Pre-emptive and Non-preemptive SJF scheduling algorithms. Explain the performance advantages and disadvantages.
- Describe the Preemptive Priority-based scheduling algorithm. Explain the performance advantages and disadvantages.
- How does the length of the time quantum affect Round-Robin scheduling?
- Which scheduling algorithms guarantee progress?
- A process was switched from running to ready state. Describe the characteristics of the scheduling algorithm being used.
- Which properties of scheduling algorithms affect the performance of interactive systems?

Semaphores / Mutexes

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