NOTE: These are interesting questions to test your general knowledge. Some might be similar to short exam questions but most exams also use longer form questions Also note that question styles can vary each term.

University of Waterloo

CS350: Final Exam Review

Gwynneth Leece, Andrew Song, Rebecca Putinski

Winter, 2010

Intro, Threads & Concurrency

- What are the three views of an operating system? Describe them.
- Define the *kernel* of the OS. What things are considered part of the OS but not the kernel?
- What are some advantages of allowing more than one thread to exists simultaneously?
- What is contained in a *thread context*? What is the *thread control block*?
- Describe what happens during a *context switch*? What is *dispatching*?
- What is the difference between `thread_yield()` and `thread_sleep()`?
- How is an involuntary context switch accomplished? Define *quantum*.
- Describe what happens when an *interrupt* occurs. Where do interrupts come from?

Synchronization

- What are the three properties of a good implementation of synchronization?
- Can the OS know when a program is in a critical section?
- When is it appropriate to disable interrupts? What happens when you disable interrupts on a multiprocessor? Give pros and cons of disabling interrupts.
- Describe *Peterson's Algorithm*. What are some restrictions on its application? Is it starvation free?
- How does *Test and Set* work? How would you use it in a program? What is a potential problem with Test and Set?
- What is a *semaphore*? Which two operations does it support? Are they atomic?
- Describe how you would synchronize producer and consumer threads using a sempahore.
- What is the primary difference between a *lock* and a *binary semaphore*?
- Is the implementation of semaphores given in the notes starvation free? Why or why not?
- What is a *condition variable*? How are they used in conjunction with locks?
- What are the differences between *Mesa-Style CVs* and *Hoare-Style CVs*?
- Describe, at a high level, what a *monitor* does. How can we simulate their behaviour in C?
- Define *deadlock*. How does the system recover from a deadlock?
- Can deadlock occur with only one thread? Why or why not?
- If there is a cycle in a resource allocation graph, is there a deadlock? What about the converse of this?
- Give the deadlock detection algorithm, and describe the three deadlock prevention methods discussed in class.

Processes and the Kernel

- What information is contained within a *process*?
- What is the difference between a *sequential process* and a *concurrent process*?
- Can threads from one process access memory from another process?
- What can the kernel do in privledged execution mode that user programs can not directly accomplish?
- Describe what happens when a system call in invoked.
- In which registers does a system call expect the system call code? Arguments?
- In which registers will the system call place the return result? Return value?
- Why does OS/161 have two stacks?
- What are the only three ways for the CPU to enter privileged mode?
- What is an *exception*? What is the difference between an exception and a system call? And exception and a interrupt?
- The `mips_trap` function is called to handle system calls, exceptions, and interrupts. How does it differentiate between these three cases?
- What does the OS/161 exception handler `common_exception` code do?
- When returning from a system call, the program counter is not automatically advanced by the hardware. Why is this?
- What might an OS want to keep track of in a *process table*?
- Define *timesharing*. When can the CPU context switch to another thread?
- What are three aspects of the process model that must be defined by the OS?

Virtual Memory: Pre-Midterm

- A system uses *physical address* and *virtual addresses*. How many physical address spaces are there in a computer? How many virtual address spaces?
- What is a simple method of address translation? How does this work? What are its disadvantages?
- In *dynamic relocation*, when does the value of the relocation register in the MMU change?
- What are three algorithms that can be used to find an allocation space? Why would anyone ever want to use the worst fit algorithm?
- What is the difference between *external fragmentation* and *internal fragmentation*?
- What advantages does *paging* give over dynamic relocation?
- How does the kernel use a page table to help translate virtual addresses to physical addresses?
- Given a virtual address and a page table, how do you calculate its physical address? In reverse?
- Describe how the OS uses the valid bit to determine which PTEs contain a valid page mapping.
- In address translation, what (4) roles does the OS serve? What (2) roles does the MMU serve?
- What happens when a process violates a protection rule checked by the MMU?
- What is the TLB? What does it stand for?

- Why might the MMU need to invalidate the TLB? When would this happen?
- Why is it not a good idea to save/restore the TLB on context switches?
- What is the difference between a hardware-controlled TLB and a software-controlled TLB? Which is the MIPS TLB?
- A range addresses starting at `0x00000000` is intentionally left empty. Why is this?
- How does *segmentation* improve address translation? What is the motivation behind segmentation?
- How do you translate a segmented and paged virtual address into a physical address? In reverse?
- Why is sharing virtual memory useful? How can this be accomplished? Give an example of when two processes should share a page.
- What are three approaches for locating the kernel's address space? How do OS/161 and Linux do this?
- Describe the different parts of an ELF file. Why doesn't the ELF file describe the stack?
- What are the different *sections* in an ELF file? What are the different *segments*?
- Know which kind of variables map to which section.
- What functions/system calls can be used to share virtual memory?

Virtual Memory: Post-Midterm

- Page tables for large address spaces may be very large. One problem is that they must be in memory, and must be physically contiguous. What are two solutions to this problem?
- What are the key differences between normal and *inverted page tables*?
- What is the difference between a *global page replacement* policy and a *local page replacement* policy?
- What steps must the OS take to handle a *page fault exception*?
- What are the 6 different page replacement policies discussed in lecture? Describe and compare them.
- What is the definition of a *compulsory fault*?
- What are the two types of *locality* to consider when designing a page replacement policy?
- In practice, frequency based page replacement policies don't work very well. Why is this?
- LRU page replacement is considered impractical in virtual memory systems. Why is this?
- Does the MIPS TLB include a *use bit*? A *modified bit*?
- Why is simulating a use bit expensive?
- Why are modified pages more expensive to replace than a clean page?
- Describe how you would simulate a modified bit in software?
- Define *page cleaning*, and explain the difference between *synchronous* and *asynchronous page cleaning*.
- What is *Belady's Anomaly*, and what is a *stack policy*? What is one way you can tell if a replacement policy is a stack policy?
- Of LRU, FIFO, and CLOCK, which algorithms are stack policies and which are not?
- What is *prefetching* and what is its goal? What are the hazards of prefetching? Which kind of locality does prefetching try and exploit?

- Give 3 advantages and 2 disadvantages of large page sizes.
- How large are the pages in OS/161?
- Describe the *Working Set Model*, *working set*, and *resident set*. Define a *phase change*.
- How can information about a program's working set be used in a page replacement policy?
- What is *thrashing* and what are two solutions to it?
- Why is it better to suspend and swap processes than just to have them all run at once?
- Describe an ideal amount of a process' working set to remain in memory in terms of suspending processes.

Processor Scheduling

- Under what circumstances would a non-preemptive scheduler run? (ie. when does a thread give up the CPU through its own actions)
- Define *CPU burst* and *I/O burst*.
- What are the 5 scheduling algorithms discussed in lecture? Describe/compare/contrast them.
- Which of those 5 algorithms is provably optimal?
- Describe how a *multilevel feedback queue* works.
- In a prioritized scheduling environment, low-priority threads risk starvation. How can this be solved?
- Describe pseudo code for such a scheduling algorithm which eliminates the risk of starvation.

I/O and Devices

- What are the four registers each device maintains? How are they used?
- Describe how SYS161 uses memory mapping for devices.
- What is the difference between *program controlled I/O* and *Direct Memory Access*? What are their respective advantages? Which does SYS/161 use?
- OS's often buffer data that is moving between devices and application programs' address spaces. What are the benefits/drawbacks of this?
- What is the only way for a user application to talk to a device?
- What does the statement "disks are non-volatile" mean?
- Using the *simplified cost model* for disk block transfer, what are the three components that make up "request service time"?
- If a disk spins at x rotations per second, what is the expected rotational latency in terms of x?
- If there are k sectors to be transferred and there are T sectors per track, what is the transfer time in terms of k, T, and x?
- If k is the required seek distance, what is the seek time (as a function of k)?
- Why is sequential I/O more efficient than non-sequential I/O?
- Define *track buffering*.
- What are the 6 disk head scheduling algorithms discussed in lecture? Who is responsible for scheduling the disk head? Describe/compare/contrast them.

File Systems

- What is the definition of a *file* given in the notes? What is the definition of a *file system*?
- Describe the 6-7 operations described in class that an application can perform on a file.
- File position may be implicit (ie. not specified on each read/write call). Why is this, and because of this how is non-sequential I/O performed?
- What happens when you seek past the end of a file?
- What does it mean for a file to be *memory mapped*? What advantages does memory-mapping give?
- What are some (4) options for how to handle a memory mapped file getting updated?
- What are the effects of the above options when concurrency is taken into consideration? (ie. what if two processes have the same file mapped)
- What is the difference between a *hard link* and a *soft link*?
- What limitations of hard links do soft links not have?
- What problems can occur with the use of soft links?
- Why can't hard links cross file system boundaries?
- If we wish to operate with multiple file systems, what are the two most common options in organizing the system (ie. the two options discussed in class)?
- What are the advantages/disadvantages of *fixed-size chunk allocation* vs. *variable-size chunk allocation*? Which system is more readily used in modern drives?
- How are drives indexed? What does LBA stand for?
- What are the 3 file indexing methods given in lecture? Describe how they work and their advantages/disadvantages.
- What kind of meta-data is containted in a Unix *i-node*?
- What are advantages of using single/double/triple indirect blocks instead of just direct blocks?
- Why is it advantageous to leave large gaps between sections of used space on disk?
- How are directories implemented?
- Describe (at a high level) how to implement hard links and soft links.
- What does the `open()` system call have to do differently when handling a symlink file?
- What are two ways for a file system to handle/clean up after a failure?

Interprocess Communication

- What are the two general methods that can be used for two processes to share information?
- Describe and compare *direct message passing* vs. *indirect message passing*.
- How can message passing mechanisms differ in terms of the following properties: directionality, message boundaries, connections, and reliability.
- Routers are allowed to drop packets. Why is this okay?
- Why don't we require routers to tell a sender when one of their packets is dropped?

- What are two common types of *sockets*, and how do they differ?
- What are two *address domains* discussed in lecture, and how do they differ?
- Given that the INET address domain can be used for processes running on the same machine and on different machines to communicate, why is the UDS domain still useful?
- With datagram sockets, when will the `recvfrom()` and `sendto()` functions block?
- With stream sockets, what is the difference between the passive process and the active process?
- With stream sockets, the active process uses the `connect()` function to send a connection request. `connect()` will block until what happens?
- With stream sockets, if the active process does not choose to bind and address to the socket, what will happen? What is the advantage of this?
- Describe *pipes* in terms of directionality, message boundaries, connections, and reliability.
- Pipes use an *implicit addressing* mechanism. How does this limit their use?
- What does the `pipe()` system call return? How is it different for a simplex pipe vs. a duplex pipe?
- Describe a *named pipe* and explain how it is different from a regular pipe. How do we restrict who is able to use a named pipe?
- Describe a *message queue*.
- What kind of communication do signals permit?
- Be familiar with the signaling diagram given in lecture (process stepping in and out of kernel space).
- What are the steps performed by the OS to send/handle a signal?
- Can a process change the default behaviour of any signal?
- Who is responsible for controlling access to the *network interface?*
- What are the 7 layers of the ISO/OSI Networking Reference Model? Which 5 are the important ones?
- Where do http, TCP/UDP, and IP fit in the above model?
- What is the *principal function* of IP?
- Describe the two internet protocols and contrast them.
- What functions do ports serve in addressing?