CS162 Fa14 Mid Term 1 Study Guide

The midterm will cover the lectures (up through 9/24), the reading specified by the lectures, the homeworks, and the connections of the concepts presented there to the initial aspects of the Project 1. With our "helical" approach to the material, you have now a broad understanding the aspects of operating systems (Processes, Address Spaces, File Systems, Sockets, Concurrency, Kernel Structure) and a fairly deep understanding of the central responsibility of the operating system - executing threads in processes, providing protection, and offering convenient shared resources. The exam will seek to affirm this understanding. It will not try to do tons of tricky, subtle programming details. You can bring a "cheatsheet" consisting of a double sided 8.5x11 sheet of paper.

Previous midterms (http://inst.eecs.berkeley.edu/~cs162/sp14/handouts.html) were later in the course and mostly emphasized caching and address space management with hand simulations of specific algorithms. In each there are a couple of useful problems. For example

s14 mid#1 - problems 4, 5, and 6
f13 mid #1 - problems 5 and 6
s13 mid 31 - problems 4 & 5 (sort of)
F12 mid#1 - problems 3&4

The questions below are intended to provide a study guide. Reviewing the course materials to extract answers to them will help integrate what you have learned so far. Those are the concepts that you will use for the rest of your life. We will have some questions like these and a few concrete code or diagram type questions.

=======

Problem: What is an operating system?

Problem: Match (by drawing a line) the hardware resource to its more convenient user abstraction and, for each, describe briefly how the application program accesses it.

Hardware Resource		User Abstraction	Mode of Access
a. Processor		Files	
b. Memory		Address Space	
c. Disk Blocks		Registers	
d. Network		Device I/O	
e. Keyboard		Signal	
f. Display		Socket	
g. Mouse		Window	

h. Interrupt		Stack	
		Lock	
		Heap	
		Semaphore	
		Thread	

Problem: The operating system provides a protection boundary to protect itself from errant or malicious applications and applications from one another. For each of the resources below describe how it is protected.

1. Kernel code: _____
2. Kernel data : _____
3. User code: _____
4. User stack: _____
5. System files: _____
6. I/O devices : _____
7. File Descriptors:
8. Process control blocks:
9. File handles
10. Locks

Problem: When a process calls `exit()`, which of the following must the kernel do?

1. Reclaim the memory allocated to the process.
2. Reclaim the process control block.
3. Close all the open files
4. Reclaim the file descriptors for the process before closing the files.
5. Reclaim the file descriptors for the process after closing the files.
6. Wait until all the threads in the process terminate.
7. Kill all the threads in the process.
8. Save the registers into the kernel stack.

Problem: Explain the difference between `fork` and `exec`.

Problem: Explain the differences between Process Fork and Thread Fork.

Problem: A web server may fork a process to handle each incoming connection. Why? Circle all that apply.

1. To isolate processing the individual connection for the rest of the server.
2. To obtain a socket for the connection that is distinct from the listening socket.
3. To put the main listening thread to sleep.
4. To process the connection concurrently with listening for additional connections.
5. To process the connection concurrently with processing other connections.
6. To overlap I/O with other processing of the connection request.

Problem: In the http server you built using process fork, the parent listening process closed the connection socket returned from accept and the child connection process closed the listening socket. Why?

If, instead, a thread were forked to handle the connection, what sockets would each thread close?

If handling the connection causes a stack overflow, how would the behavior be different in the two styles of server design?

Problem: When an interrupt occurs during execution of a user thread, which of the following are performed by the hardware and which by the operating system? (Some may not be performed by either.) If done by the OS, indicate whether it is always or sometimes performed.

1. Transfer control to the interrupt handler specified in the interrupt vector.
2. Save the user registers onto the user stack
3. Disable interrupts
4. Enable interrupts
5. Execute a return-from-interrupt instruction
6. Set system mode
7. Service the interrupt
8. Open a file
9. Terminate the user thread
10. Switch to another thread
11. Resume the user thread

Problem: How is a syscall different from an interrupt?

Problem: When a thread is executing on a processor, which of the following are true?

1. PC register holds the address of the executing instruction
2. Registers hold the current root state of the thread
3. Registers hold the entire state of the thread
4. Registers hold the state of the kernel
5. The stack pointer is valid
6. A kernel thread is waiting for it
7. The thread is on the ready queue.
8. The thread is marked BLOCKED
9. The thread is marked RUNNING
10. Interrupts are disabled
11. It holds locks.

Problem: What segments of an executing process are contained in the executable file for the application and which are created upon loading?

Problem: What is minimally needed to support dual-mode operation?

Problem: How does the operating system prevent one user process from accessing the code and data of another user process?

Problem: How does the kernel create a separate stack for each kernel thread?

Problem: What is the role of the scheduler?

Problem: Describe the structure and role of each of the following namespaces. For each, indicate whether it is implemented in the hardware, the operating system, or the library in the application.

1. Process ID
2. Memory Address Space
3. Virtual Address Space
4. File names
5. File streams
6. File handles
7. Hostnames
8. URLs
9. Port numbers
10. Thread IDs
11. Socket descriptors

Problem: What are the advantages of the Unix design philosophy that treats permanent storage, device I/O, and inter-process communication uniformly?

Problem: What are the benefits of the open-before-use model of file access?

Problem: How does the file abstraction in modern operating systems include aspects of the process abstraction? How does it include aspects of concept of users?

Problem: What aspects of protection arise in file systems?

Problem: What is the division of responsibilities between the user I/O library, the file system, and the I/O driver?

Problem: When a user thread issues a read syscall, how does control get transferred to the read operation in the file system?

Problem: Explain the role of the file position in the high level and low level I/O abstractions.

Problem: The basic read operation is blocking, i.e., it returns when the data to be read is transferred into the buffer. What does the OS do with the user thread while the read takes place?

Problem: Using the basic pthread operations, show how a user process could continue processing while a file read is outstanding. At what point is it valid to access the data in the read buffer?

Problem: Both files and sockets are accessed by writes and reads on a descriptor handle. How does "opening" a socket differ from opening a file?

Problem: Can files be used as a means of communication between two processes on the same machine?

Problem: Within a single sequential process, i.e., a thread, execution follows a sequential flow of control. However, with multiple threads or processes, they may proceed at independent rates and some kind of rendezvous is requires to ensure proper ordering.

- Explain what happens if a client performs a connect before the server it is trying connect to performs a listen.
- Explain what happens if a server performs a listen before any client connects to it?
- How far must the server get in its socket creation process before clients can connect to it without getting an error?

Problem: explain the role of each of Bind, Listen, and Accept in creating a server socket. How do these differ from Connect?

Problem: When a server listens on a socket, what does the operating system do with its thread?

Problem: Describe the lifecycle of a process as a state machine. Give one concrete example of what might cause each state transition.

Problem: What information would need to be in a Process Control Block that is not in thread control block.

Problem: Show how a Lock can be implemented using a semaphore with operations sema_init, sema_up, and sema_down.

Problem: How does the address space of a multithreaded process differ from that of a single threaded process? What is required in the kernel to support multithreaded processes beyond that required to support single-threaded processes?

Problem: What does the operating system do with a thread that executes a thread_join?

Problem: What are the similarities and differences between Lock Acquire and Thread Join?

Problem: What are the similarities and differences between Lock Release and Thread Exit?

Problem: What thread may execute immediately following a thread_yield? (All that apply)

1. The thread performing the yield.
2. The parent thread of that performing the yield.
3. The child of the thread performing the yield.
4. A thread waiting on this thread.
5. The thread that has performed a thread join on the yielding thread.
6. Any thread.
7. Any thread that is READY when the yield is performed.
8. A thread waiting on a lock held by the yielding thread.

Problem: How can thread_yield be used to correctly synchronize cooperating threads or processes?

Problem: How is an interrupt hander different from a thread?

Problem: What state or information is maintained in the kernel for each thread?

Problem: When a kernel thread that provides the system processing associated with a single-threaded user process resumes that process, what is the state of the kernel thread? What resumes the kernel thread?

Problem: What does an interrupt handler need to do in order to save the state of the thread that it interrupted?
Problem: Why must interrupts be disabled when a kernel thread enters thread_switch? What re-enables them?

Problem: Describe conditions under which FIFO has lower average response time than Round-Robin scheduling? Described conditions where Round-Robin has lower average response time.

Problem: Using semaphores, rather than a flag or locks, implement a 1-1 producer consumer relationship. Generalize your solution to allow k consumers. (Actual question would provide specific code fragments for you to work with.)

Problem Schema: Given a diagram of a process address space and kernel data structures, show the change to the diagram after a user thread folk, a user process fork, a kernel thread create, or other such operations.

Problem Schema: Show (or fix) how to create a certain specific threadsafe data structure.

Problem Schema: Show (or fix) how to make some legacy library functions threadsafe.