CMSC 421 Midterm Answers
Contributions: Corey Atkins, Mena Beshai Veronica Clements, Max Poole,

1. **What's the difference between a duck?**
   Freebie

2. **What is the difference between code running in user mode versus kernel mode? How is this supported by the hardware in modern processors?**
   Kernel Mode: kernel code executes in processor's privileged mode (kernel mode) with full access to all physical resources of computer.
   Any OS-support code that doesn't need to run in kernel mode is placed into system libraries and runs in **user mode**
   Under Linux, no user code is built into kernel
   (pg 787)

3. **One way in which a processor switches from user mode to kernel mode is when user code executes a system call. What are two other ways?**
   Hardware Interrupts (timers, keyboard, I/O) to execute interrupt handler in kernel mode
   [software] Exception → software interrupt caused by illegal action, e.g. dividing by 0
   [software] Trap → software interrupt done on purpose, e.g. a system call
   //Runtime Error/crashing.(Wrote this, got full points.)
   //Exceptions (had this written on exam)

   (http://www.tldp.org/HOWTO/KernelAnalysis-HOWTO-3.html)
(https://stackoverflow.com/questions/1311402/differences-between-user-and-kernel-modes)

4. **What does it mean when we say an operating system is "interrupt driven?"**
   OS only runs after interrupts

5. **What is the difference between a simple library call versus a system call? Give two examples of system calls you might typically find in a modern OS.**

   System Calls: programming interface to services provided by OS
   - Handled by kernel (kernel mode)

   Library Calls: include ANSI C standard library and are portable; linked to application programs; always executes in user mode
   (http://ksearch.wordpress.com/2010/09/12/system-call-vs-library-function)

6. **What is the primary difference using multithreading and multiprocessing?**

   Multithreading: being executed in 1 process sharing common address space (more than 1 thread executed parallely on 1 CPU).

Multiprocessing: different processes have different address space (uses more than 1 CPU)
(https://www.youtube.com/watch?v=oIN488Ldg9k)
(https://www.thecrazyprogrammer.com/2014/12/difference-between-multiprogramming-multitasking-multiprocessing-multithreading.html)

7. **What is the motivation of minimizing the sizes (in number of instructions) of the locked critical sections of a program or OS?**
   The motivation of minimizing sizes is because no other processes can be in their critical section while another process is in its critical section. Minimization of these sizes can in turn move other processes forward in implementing their critical sections. We just want to be out of the critical section ASAP because they are exclusive.

   (wrote on exam, will attempt to find source soon)

8. **Explain what is involved in a context switch: when does it occur, and what exactly are the context elements that must be switched?**
   Context Switch: switching CPU to another process requires state save of current process and state restore of different processes. Occurs when kernel transfers control of CPU from executing process from another ready to run.

   (pg 114)

9. **The first version of the consumer-producer problem that we studied had data structures -- the buffer, in, and out variables -- that were being simultaneously accessed by two processes. Why was this guaranteed safe even though no locking was used?**

The producer only updated count variables that the consumer used, and the consumer only updated count variables that the producer used. Each process does a busy wait until variables were correct.

10. **Compare interprocess communication mechanism based on shared memory versus message-passing models.**

    Message-passing models: processes communicate with each other without resorting to shared variables (also known as communication by means of messages)
    Shared memory: region of memory that is shared by cooperating processes established
    - Processes can exchange information by reading/writing data to shared region

    Message passing: smaller amount of data, easier to implement in distributed system.
    Shared memory: faster, no assistance from kernel

(pg 122)

11. **Give an example in pseudocode or a race condition that will potentially result in an incorrect update to a variable that is shared between two processes.**

```
Counter = 5;
Do{
        Counter++;
}while(true);

Counter = 5;
Do{
        Counter--;
}while(true);
```

12. **What are the three necessary properties for a solution to the Critical Section Problem?**

    Mutual Exclusion: If process Pi executing in its critical section, than no other processes can be executing in their critical sections

    Progress: If no process is executing in its critical section and there exists some other processes that wish to enter their critical section, then the selection of the processes that will enter critical section next cannot be postponed indefinitely.

    Bounded Waiting: Bound must exist on the time a process must wait after process has made request to enter its critical section and before request granted.

    (pg 206-207)

13. **We talked about two common atomic operations available on modern CPUs. Name one of them and give a brief description; make sure to explain why it is important.**
    Atomic operations -- unable to to change/uninterruptible
    Important in order to swap contents of two words in multiprocessor systems
    Test_and_set: do while true (useful for obtaining locks)
    Compare_and_swap

    (pg 210-211)

14. **Show how deadlock can occur between two processes that share a set of locks (Assume you have two mutex locks, m1 and m2, and that you use the primitive wait(mutex) and signal(mutex) to acquire and release these locks).**

. . .	. . .
wait(m1);	wait(m2);
wait(m2);	wait(m1);
. . .	. . .

15. **What kind of access patterns is a solution to the readers-writers synchronization problem supposed to support? (Explain in terms of permitted/prohibited concurrent operations).**

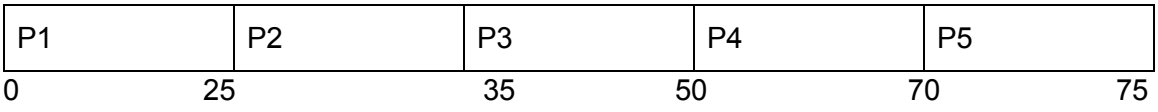
It should permit concurrent read access but prohibit any concurrent access when writing is occuring. Essentially multiple readers, 1 writer, never both at once.
(pg 220)

16. **What are two features that are gained in moving up from a classic mutex to a semaphore?**
    A semaphore allows for access of a resources by multiple different processes at once.
    Can implement without a spinlock

17. **Given the above list of processes arriving at time t= on the order shown:**

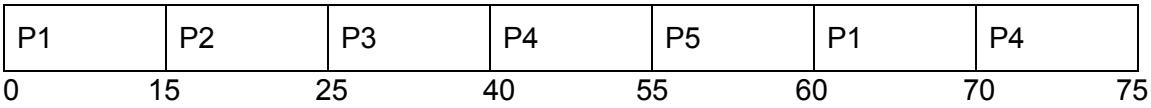
Process	**Distribution**
P1	**25**
P2	**10**
P3	**15**
P4	**20**
P5	**5**

a. **Draw a Gantt chart showing how the processes would be scheduled by first-come/first-serve (FCFS) scheduling algorithm. In your chart, label the exact times when each process would be switched in. What is the average wait time with FCFS?**



Average time = 0+25+35+50+70/ 5 = 36

b. Using the same data, draw a Gantt chart for what a preemptive Round Robin scheduler with a quantum of 15ms would do. What is the average wait time with this strategy?



Average time = (0+(60-15)) + 15 + 35 + 40 + (75-55) + 55/5 = 39

18. The simple FCFS scheduling algorithm is subject to the convoy effect--describe what that is and why it is bad.

The Convoy Effect happens when a few slow processes slow down the whole system. This is bad because in the same time we were servicing the couple of slow processes, we could have been servicing many faster processes. This means we are not making efficient use of our CPU cycles and will lead to poor performance.

(pg 267)

19. What is the difference between hard real-time systems and soft real-time systems?

In general a real time system sets a deadline on when an operation must complete. A hard real-time and soft real-time differ in how hard of a deadline that is. In a hard real-time system missing a deadline leads to complete system failure. In a soft real time system missing a deadline only leads to a degradation of the usefulness of the result.
https://stackoverflow.com/questions/17308956/differences-between-hard-real-time-soft-real-time-and-firm-real-time

20. The semantics for a fork() call are that the resulting processes have identical copies of everything, including the call stack. By contrast, in multithreading, the two threads must have their own stack at different addresses--why?

Threads need to have their own local variables that are not put into shared memories; otherwise, they would clash into each other unexpectedly. Fork() doesn't have that problem as we should call exec which will replace the memory contents.

21. Describe two different methods of evaluating scheduling algorithms.

Two different methods of evaluating scheduling algorithms include the deterministic and simulation methods. In the deterministic method we give each algorithm a predetermined workload, each algorithm is then evaluated according to that workload. In a simulation we actually try to simulate a whole machine. We can then gather information at each clock tick and use that to evaluate our scheduling algorithms.

(pg 300-303)

EXTRA CREDIT: PETERSON'S SOLUTION TO THE CRITICAL SELECTION PROBLEM (PSUEDO-CODE)

```
Do{
        flag[i]==true;
        Turn = j;
        while(flag[i] && turn == j);
                Critical section;
        Flag[i] = false;
                Remainder section;
}while(true);
```