# Midterm

CS 414 Operating Systems, Spring 2007
March 8th, 2007
Prof. Hakim Weatherspoon

Name: ________________________________NetId/Email:__________________________

**Read all of the following information before starting the exam:**
Write down your name and NetId/email NOW.

This is a **closed book and notes** examination. You have 90 minutes to answer as many questions as possible. The number in parentheses at the beginning of each question indicates the number of points given to the question; there are 100 points in all. You should read **all** of the questions before starting the exam, as some of the questions are substantially more time consuming.

Write all of your answers directly on this paper. ***Make your answers as concise as possible.*** If a question is unclear, please simply answer the question and state your assumptions clearly. If you believe a question is open to interpretation, then please ask us about it!

**Good Luck!!**

| Problem | Possible | Score |
|---|---|---|
| **1** | 26 | |
| **2** | 24 | |
| **3** | 16 | |
| **4** | 12 | |
| **5** | 22 | |
| **Total** | **100** | |

1. (26 points total) Short answer questions (***NO answer should be longer than two or three sentences***).

a. (2 points) Name three ways in which the processor can transition form user mode to kernel mode? Can the user execute arbitrary code after transition?

b. (6 points) Threads

i) (2 points) What needs to be saved and restored on a context switch between two threads in the same process? What if two are in different processes? Be brief and explicit.

ii) (2 points) Why is switching threads less costly than switching processes?

iii) (2 points) Suppose a thread is running in a critical section of code, meaning that it has acquired all the locks through proper arbitration. Can it get context switched? Why or why not?

c. (6 points) Deadlock

i) (2 points) Name the four conditions required for deadlock and give a brief (one sentence) description of each.

ii) (2 points) Does a cyclic dependency always lead to deadlock? Why or why not?

iii) (2 points) What is the difference between deadlock prevention and deadlock avoidance? What category does Bankers algorithm falls in and why?

d. (2 points) What are exceptions? Name two different types of exceptions and give an example of each type.

e. (2 points) Why would two processes want to use shared memory for communication instead of using message passing?

f. (2 points) What is internal fragmentation? External fragmentation? Give a brief example of each.

g. (6 points) For each of the following thread state transitions, say whether the transition is legal *and* how the transition occurs or why it cannot. Assume Mesa-style monitors.

i) (2 points) Change from thread state BLOCKED to thread state RUNNING

ii) (2 points) Change from thread state RUNNING to thread state BLOCKED

iii) (2 points) Change from thread state RUNNABLE to thread state BLOCKED

***EXTRA CREDIT***

h. (3 points) Suppose you have a concurrent system with locks: Lock.acquire() blocks until the Lock is available and then acquires it. Lock.release() releases the Lock. There is also a Lock.isFree(), that does *not* block and returns true if the Lock is available; otherwise, returns false .

What can you conclude about a subsequent Lock.acquire(), based on the result of a previous call to Lock.isFree() ?

2. (24 points total) CPU Scheduling. Here is a table of processes and their associated arrival and running times.

| **Process ID** | **Arrival Time** | **Expected CPU Running Time** |
|---|---|---|
| Process 1 | 0 | 5 |
| Process 2 | 1 | 5 |
| Process 3 | 5 | 3 |
| Process 4 | 6 | 2 |

a. (12 points) Show the scheduling order for these processes under First-In-First-Out (FIFO), Shortest-Job First (SJF), and Round-Robin (RR) with a quantum = 1 time unit. *Assume that the context switch overhead is 0 and new processes are added to the* ***head*** *of the queue except for FIFO.*

| **Time** | **FIFO** | **SJF** | **RR** |
|---|---|---|---|
| **0** | | | |
| **1** | | | |
| **2** | | | |
| **3** | | | |
| **4** | | | |
| **5** | | | |
| **6** | | | |
| **7** | | | |
| **8** | | | |
| **9** | | | |
| **10** | | | |
| **11** | | | |
| **12** | | | |
| **13** | | | |
| **14** | | | |
| **15** | | | |

b. (12 points) For each process in each schedule above, indicate the queue wait time and turnaround time (TRT).

| Scheduler | Process 1 | Process 2 | Process 3 | Process 4 |
|---|---|---|---|---|
| FIFO queue wait | | | | |
| FIFO TRT | | | | |
| SJF queue wait | | | | |
| SJF TRT | | | | |
| RR queue wait | | | | |
| RR TRT | | | | |

The queue wait time is the *total* time a process spends in the wait queue.
The turnaround time is defined as the time a process takes to complete after it first arrives.

3. (16 points) Virtual Memory Page Replacement
Given the following stream of page references by an application, calculate the number of page faults the application would incur with the following page replacement algorithms. Assume that all pages are initially free.

Reference Stream: A B C D A B E A B C D E B A B

a. (4 points) FIFO page replacement with 3 physical pages available.

b. (4 points) LRU page replacement with 3 physical pages available.

c. (4 points) OPT page replacement with 3 physical pages available.

d. (4 points) True or False. If we increase the number of physical pages from 3 to 4, the number of page faults always decreases using FIFO page replacement. Briefly explain.

4. (12 points) Memory Management

a. (6 points) Consider a memory system with a cache access time of 10ns and a memory access time of 200ns. If the *effective access time* is 10% greater than the cache access time, what is the hit ratio H? (Fractional answers are okay).

b. (6 points) Assuming a page size of 1 KB and that each page table entry (PTE) takes 4 bytes, how many levels of page tables would be required to map a *34-bit* address if every page table fits into a single page. Be explicit in your explanation.

5. (22 points) Concurrency: the "$H_2O$" problem
You have just been hired by Mother Nature to help her out with the chemical reaction to form water, which she does not seem to be able to get right due to synchronization problems. The trick is to get two H atoms and one O atom all together at the same time. The atoms are threads. Each H atom invokes a procedure *hReady* when it is ready to react, and each O atom invokes a procedure *oReady* when it is ready. For this problem, you are to write the code for *hReady* and *oReady*. The procedures must delay until there are at least two H atoms and one O atom present, and then one of the threads must call the procedure *makeWater* (which just prints out a debug message that water was made). After the *makeWater* call, two instances of *hReady* and one instance of *oReady* should return. Your solution should avoid starvation and busy-waiting.

You may assume that the semaphore implementation enforces FIFO order for wakeups—the thread waiting longest in P() is always the next thread woken up by a call to V().

a. (4 points) Specify the correctness constraints. Be succinct and explicit.

b. (18 points) provide the pseudo implementation of *hReady* and *oReady* using semaphores.