University of Pittsburgh
CS/COE 1550: Introduction to Operating Systems — Midterm Exam
Dr. Daniel Mosse — March 5, 2020

Instructions:

- please write your PeopleSoft ID on top of every page
- You have 75 minutes to solve all the questions in the exam.
- **The exam will be graded out of 72 points (i.e., there are 10 bonus points)**.
- The exam is closed-book and closed-notes. You can use a basic calculator.
- Please use *a pen or a dark pencil*.
- Answer each question in the scantron/bubble sheets (T/F and Multiple Choice) or the boxes/space provided. If you need extra space use additional sheets. Remember to write your name and problem number on any additional sheets you use.
- Plan your time wisely. It is recommended to read all problems through first. Do not spend too much time on any single problem. Make sure that your answers are clear and neat.

Question:	1	2	Total
Points:	48	34	82
Score:			

Name: _____ PeopleSoft ID: _____

Question 1

(a) (16 marks) **True/False ON SCANTRON/BUBBLE SHEETS**

1. ☐ Threads of the same process share their entire address space, that is, they have no private portions of the address space.
2. ☐ If interrupts are disabled, preemption is not possible.
3. ☐ A CPU-bound process also contains input/output operations.
4. ☐ To add a new system call, you do not change the operating system source code, just have to add the definitions in a library.
5. ☐ A process's CPU burst time is always equal to the quantum.
6. ☐ Semaphores are the only mechanism to synchronize threats or processes.
7. ☐ In a single-core system, it is safe to assume that mutual exclusion is guaranteed when interrupts are disabled around a critical section.
8. ☐ A process that accesses only one semaphore can be involved in a deadlock

(b) (14 marks) **Multiple Choice ON SCANTRON/BUBBLE SHEETS**

1. Examples of short-term scheduling policies are:
   - ◯ Round-Robin and FCFS
   - ◯ Non-primitive and Shortest Remaining Time First
   - ◯ Preemptive and Shortest Job First
   - ◯ none of the above
2. A job is in the ready queue if it:
   - ◯ has all resources needed to execute
   - ◯ is waiting for an interrupt
   - ◯ is waiting for I/O
   - ◯ has all resources needed to execute, except CPU
3. A process is preempted by the kernel:
   - ◯ after the quantum expires
   - ◯ when a process gives up the CPU voluntarily
   - ◯ when a process requests I/O
   - ◯ when a process is killed
4. Shortest job first scheduling policy that considers new processes/jobs may be added to the system:
   - ◯ is a fair scheduling policy
   - ◯ favors long jobs
   - ◯ may cause starvation of some processes
   - ◯ is optimal
5. A binary semaphore can have values
   - ◯ a. zero or one
   - ◯ b. zero or any positive number
   - ◯ c. zero or -1
   - ◯ a. and b. only
   - ◯ a. and c. only
6. (*Mark all that apply*) Under what conditions are spinlocks efficient?
   - ☐ single-core systems
   - ☐ multi-core systems with few threads competing for the lock
   - ☐ multi-core systems with many threads competing for the lock
   - ☐ short critical section
7. (*Mark all that apply*) The two **mechanisms** for scheduling are:
   - ☐ Barrier synchronization
   - ☐ Mutual exclusion
   - ☐ Hold and wait
   - ☐ Preemption

(c) **Short Answer**

1. (2 marks) How and where does the kernel keep track of CPU burst length of a process?

2. (2 marks) Give a formula for context switch overhead in a preemptive system.

3. (2 marks) Explain why it makes sense to give an I/O-bound process higher priority for the CPU than a CPU-bound process.

4. (2 marks) What can cause a process's state to change from "running" to "ready"?

5. (2 marks) What are the registers needed for virtual to physical address translation in contiguous memory allocation?

6. (2 marks) Give a formula or describe briefly how virtual to physical address translation happens?

7. (2 marks) What are two ways to keep track of free memory in contiguous memory allocation?

8. (4 marks) Consider a system with two or more processes that use semaphores. Show pseudo-code for processes in deadlock.

Total for Question 1: 48 marks

Question 2

(a) **Deadlock Avoidance.** Consider the following snapshot of a system that has four resource types: A, B, C, and D and five processes, P0, P1, P2, P3, and P4.

	Allocation	Max	Available
	A B C D	A B C D	A B C D
P0	2 0 0 0	4 2 1 2	3 3 3 1
P1	3 1 2 1	5 2 5 2	
P2	2 1 0 3	2 3 1 6	
P3	1 3 1 2	1 4 2 4	
P4	1 4 3 2	3 6 6 5	

Answer the following questions using the banker's algorithm:

1. (4 marks) Illustrate that the system is in a safe state by demonstrating an order in which the processes may complete. Write the final process order in the box below. **Show all steps.**

Process order:

2. (2 marks) If a request from process P1 arrives for (1, 1, 0, 0), can the request be granted immediately? Why? Write the final process order or the word None if no finishing order exists in the box below. **Show all steps.**

Process order:

3. (2 marks) If a request from process P4 arrives for (0, 0, 2, 0), can the request be granted immediately? Why? Write the final process order or the word None if no finishing order in the box below. **Show all steps.**

Process order:

(b) **CPU Scheduling Policies.** Consider the following set of processes, with the length of the CPU burst and the arrival time given in milliseconds.

Process	Burst Time	Priority	Arrival Time
P1	1	1	3
P2	5	3	2
P3	2	5	0
P4	8	4	1
P5	4	2	5

1. (10 marks) Calculate the average turnaround time (finish time minus arrival time) for each of the following scheduling policies. **Show all steps, state assumptions,** and **write your final results inside the boxes.**

a. [3] Non-Preemptive Shortest Job First

Average turnaround time = [ ]

b. [3] Round Robin (quantum = 2)

Average turnaround time = [ ]

c. [4] Multilevel feedback queues with three queues: from highest to lowest priority, q_0 (quantum = 2), q_1 (quantum = 4), and q_2 (quantum = ∞).

Average turnaround time = ____

(c) (6 marks) **Memory Allocation** Describe all the steps pertaining to memory allocation in contiguous memory system, from process creation to process termination. Included in your answer who does what (e.g., operating system, user level process, MMU, ...).

(d) (10 marks) **Synchronization/Semaphore Problem: forming water molecules**
The trick is to get two H atoms and one O atom all together at the same time. The atoms are threads/processes. Each H atom invokes a procedure hReady when it's ready to react, and each O atom invokes a procedure oReady when it's ready. For this problem, you are to write the code for hReady and oReady. The procedures must wait until there are at least two H atoms and one O atom present, and then one of the procedures 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. Write the code for hReady and oReady using either semaphores or locks and condition variables for synchronization. Your solution must avoid starvation and busy-waiting.

Total for Question 2: 34 marks