Exam 1

601.418/618 Operating Systems

February 26, 2024

Complete all questions.

Time: 75 minutes.

I affirm that I have completed this exam without unauthorized assistance from any person, materials, or device.

Signed: ____________________

Print name: ____________________

Date: ____________________

Question 1. [10 points] For each of the following operations, state whether it should be a privileged or unprivileged operation, and briefly justify your choice.

Operation	Privileged or unprivileged?	Justification
Store a value to a general-purpose CPU register		
Store a value to a location in virtual memory		
Store a value to a location in physical memory		
Register a handler for a hardware interrupt		
Read data from the keyboard controller		
Disable interrupts		
Enable interrupts		
Modify a page table entry		
Write data to the disk controller		
Flush the TLB		

Question 2. [10 points] Recall that the five process states are Created, Ready, Running, Waiting, and Finished. (For this question, we will assume that each process has a single thread.)

(a) Draw a diagram showing the states and transitions between states. Each transition should be drawn as an arrow showing its direction.

(b) Give an example of a situation in which a process will transition from the Running state to the Waiting state.

(c) Give an example of a situation in which a process will transition from the Waiting state to the Ready state.

(d) Is it possible for a process to transition from the Running state to the Ready state? If so, give an example of a situation where that would happen. If not, explain why not.

Question 3. [10 points]

Processes:

Process	Arrival	CPU time
0	1	8
1	4	7
2	7	2
3	8	4

Definitions:

The *wait time* of a process is
$T_{start} - T_{arrival}$

The *turnaround time* of a process is
$T_{finish} - T_{arrival}$

(a) Determine the start time, finish time, wait time, and turnaround time of each process assuming *First Come, First Served* (FCFS) scheduling. Assume that once a process begins executing it continues until it has finished. Drawing a diagram will be helpful (you can use to opposite page for scratch work if necessary.)

Process	Start time	Finish time	Wait time	Turnaround time
0				
1				
2				
3				

(b) What is the average wait time and average turnaround time for the above processes using FCFS scheduling? You may express your answer as a fraction.

Average wait time:

Average turnaround time:

[Use this page for scratch work if necessary for Question 3.]

Question 4. [10 points]

Processes:

Process	Arrival	CPU time
0	1	8
1	4	7
2	7	2
3	8	4

Definitions:

The *wait time* of a process is
$T_{start} - T_{arrival}$

The *turnaround time* of a process is
$T_{finish} - T_{arrival}$

(a) Determine the start time, finish time, wait time, and turnaround time of each process assuming *Shortest Job First* (SJF) scheduling. Assume that once a process begins executing it continues until it has finished. Drawing a diagram will be helpful (you can use to opposite page for scratch work if necessary.)

Process	Start time	Finish time	Wait time	Turnaround time
0				
1				
2				
3				

(b) What is the average wait time and average turnaround time for the above processes using SJF scheduling? You may express your answer as a fraction.

Average wait time:

Average turnaround time:

[Use this page for scratch work for Question 4.]

Question 5. [10 points]

Processes:

Process	Arrival	CPU time
0	1	8
1	4	7
2	7	2

Definitions:

The *wait time* of a process is
$T_{start} - T_{arrival}$

The *turnaround time* of a process is
$T_{finish} - T_{arrival}$

(a) Determine the start time, finish time, wait time, and turnaround time of each process assuming *Round Robin* (RR) scheduling. Assume that the scheduling quantum is 1 (i.e., time slices are one unit of time in duration.) Also, assume that if a process's arrival time is k, it starts just *before* time k. (For example, if a process arrives at time 3, it is ready to execute just *before* the time slice starting at time 3.) Drawing a diagram will be helpful (you can use to opposite page for scratch work.)

Process	Start time	Finish time	Wait time	Turnaround time
0				
1				
2				

(b) What is the average wait time and average turnaround time for the above processes using RR scheduling (with a quantum of 1)?

Average wait time:

Average turnaround time:

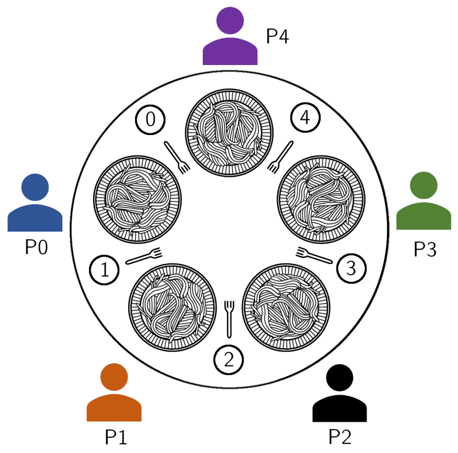
[Use this page for scratch work for Question 4.]

Question 6. [4 points] Briefly explain why the timer interrupt is necessary.

Question 7. [3 points] Briefly discuss the relative advantages and disadvantages of a shorter scheduling quantum vs. a longer scheduling quantum.

Question 8. [3 points] Briefly discuss the relative advantages and disadvantages of the M:N thread model vs. the 1:1 thread model.

Question 9. [30 points] Recall that in the dining philosophers problem, N philosophers sit at a table with N forks. Each philosopher requires two forks to eat, and must obtain both forks (to her left and right) before eating. Each philosopher is described by the following code:



```
int i = /* philosopher number, in
           range 1..N, inclusive */;

while (true) {
  think();
  obtain_forks(i);  /* acquire forks i and (i+1)%N */
  eat();
  release_forks(i); /* release forks i and (i+1)%N */
}
```

(a) Assume that the forks are represented as an array of semaphore with initial count 1, and assume `obtain_forks()` and `release_forks()` are implemented as follows:

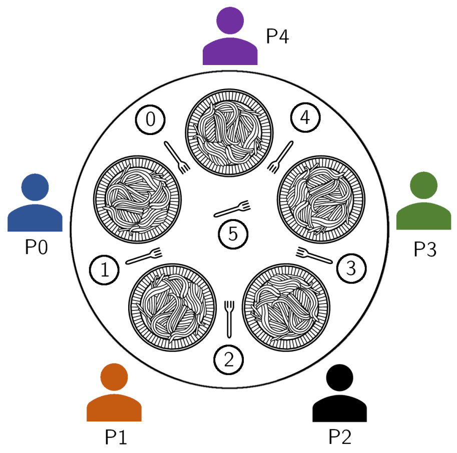
```
#define N 5           /* number of philosophers */
semaphore forks[N]; /* each initialized to 1 */

void obtain_forks(int i) {        void release_forks(int i) {
  down(&forks[i]);                  up(&forks[i]);
  down(&forks[(i+1) % N]);          up(&forks[(i+1) % N]);
}                                 }
```

Describe a scenario where the system reaches deadlock.

[Question 9 continues on next page.]

[Question 9 continues.]

(b) Consider a modified setup where in addition to the N forks positioned between the philosophers, there is one additional fork in the center of the table, numbered N, for a total of $N + 1$ forks. Assume that in addition to being able to reach the forks to her left and right, each philosopher can reach the fork in the center of the table, and that acquiring any two forks is sufficient to be able to eat.



(a) Show implementations of the `obtain_forks()` and `release_forks()` functions which are both *fair* and not prone to deadlock.

```
#define N 5            /* number of philosophers */
semaphore forks[N+1]; /* each initialized to 1 */
```

```
 void obtain_forks(int i) {        void release_forks(int i) {
```

(b) Briefly explain why your solution is free from deadlocks.

(c) Briefly explain why your solution is *fair*. ("Fair" means that no task has any advantage or disadvantage in gaining access to resources compared to other tasks.) Assume that semaphores are fair (meaning that `down` operations proceed in order.)

Question 10. [10 points]

(a) In a "traditional" operating system in which each process has (effectively) a single thread, what information is part of the *Process Control Block* (PCB)?

(b) In an operating system using the 1:1 thread model, what information is part of the *Thread Control Block* (TCB)? Hint: it should be a subset of what you listed in (a).

[Extra page for answers and/or scratch work.]

[Extra page for answers and/or scratch work.]