

CS 304 : Mid Semester Examination			Marks obtained ↓
Date: 26.02.2020,	Total questions: 8	Total Marks: 35	
Name:	Roll No:	Time: 2 hrs	

Question:	1	2	3	4	5	6	7	8	Total
Points:	5	3	5	6	4	2	5	5	35
Score:									

Instructions:

1. Write down your name and roll no. at the top of every page.
2. Please make any assumptions that you deem to be reasonable.
3. Every answer needs to be written neatly and the final answer needs to be written with a pen.
4. You are not allowed to carry any electronic gadgets including calculators and mobile phones.

- 5 1. (A) What is the difference between a mode switch and a context switch?
- (B) Most operating systems are designed for general-purpose computation. A proposal has been put forth for an OS that is optimized for running math-intensive programs. In MathOS, the kernel includes system calls for many useful mathematical operations, such as matrix arithmetic, Bessel functions, Euclidean distance, etc. These system calls are written in highly optimized assembly language for maximum performance. Is this concept for MathOS a good idea? Explain why or why not. *yes*
- (C) List two events that may take a process to a ready state.
- (D) A short quantum allows a scheduler to cycle through more processes more quickly than with a long quantum. What is the downside of this?
- (E) Given that we can create user-level code to control access to critical sections (e.g., Peterson's algorithm), why is it important for an operating system to provide synchronization facilities such as semaphores in the kernel?

- 3 2. How many times does the following program print hello? Justify your answer.

```

1 #include <stdio.h>
2 #include <unistd.h>
3 main()
4 {
5     int i;
6     for (i=0; i<3; i++)
7         fork();
8     printf("hello\n");
9 }

```

- 5 3. Given the following set of processes, with arrival time and length of the CPU-burst time given in milliseconds: For each of the following scheduling algorithms, construct the Gantt chart depicting the sequence of process execution would result on an uni-processor and calculate the average waiting time of each algorithm

- (A) First-Come First-Serve
- (B) Shortest Job First

Process	Arrival Time	Burst Time
P1	0	4
P2	2	12
P3	5	2
P4	6	6
P5	8	10
P6	12	3
P7	15	8
P8	22	5

(C) Round Robin (time quantum = 6ms)

(D) Shortest Remaining Time First

- 6 4. Assume you have a system with three processes (X, Y, and Z) and a single CPU. Process X has the highest priority, process Z has the lowest, and process Y is in the middle. Assume a priority-based scheduler (i.e., the scheduler runs the highest priority job, performing preemption as necessary). Processes can be in one of five states: RUNNING, READY, BLOCKED, not yet created, or terminated. Given the following cumulative timeline of process behavior, indicate the state the specified process is in AFTER that step, and all preceding steps, have taken place. Assume the scheduler has reacted to the specified workload change. For all the parts in this question, use the following options for each answer: a) RUNNING b) READY c) BLOCKED d) Process has not been created yet e) Not enough information to determine OR None of the above. Also, explain the reason behind.

(A) Process X is loaded into memory and begins; it is the only user-level process in the system. Process X is in which state?

(B) Process X calls `fork()` and creates Process Y. Process X is in which state?, Process Y is in which state?

(C) The running process issues an I/O request to the disk. Process X is in which state?, Process Y is in which state?

(D) The running process calls `fork()` and creates process Z. Process X is in which state?, Process Y is in which state? and Process Z is in which state?

(E) The previously issued I/O request completes. Process X is in which state? Process Y is in which state? Process Z is in which state?

(F) The running process completes. Process X is in which state?, Process Y is in which state? and Process Z is in which state?

- 4 5. Designate if the statement is True or False and explain the reason

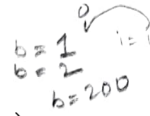
- (A) Threads that are part of the same process share the same stack. *cache.*
- (B) Threads that are part of the same process can access the same TLB entries.
- (C) A lock implementation should block instead of spin if it will always be used only on an uniprocessor.
- (D) On a multiprocessor, a lock implementation should block instead of spin if it is known that the lock will be available before the time required for a context-switch.
- (E) Peterson's algorithm uses the atomic fetch-and-add instruction to provide mutual exclusion for two threads.
- (F) When a thread returns from a call to `cond_wait()` it can safely assume that it holds the corresponding mutex.
- (G) With producer/consumer relationships and a finite-sized circular shared buffer, producing threads must wait until there is an empty element of the buffer.
- (H) Deadlock can be avoided by using semaphores instead of locks for mutual exclusion.

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

(B) Briefly explain the three conditions that a solution to a Critical Section problem must hold

- 5 7. For this question, assume the following code is compiled and run on a modern linux machine (assume any irrelevant details have been omitted):

```
1 volatile int balance = 0;
2 void *mythread(void *arg){
3     int i;
4     for (i= 0; i< 200; i++) {
5         balance++;
6     }
7     printf("Balance is %d\n", balance);
8     return NULL;
9 }
10 int main(int argc, char *argv[]) {
11     pthread_t p1, p2, p3;
12
13     pthread_create(&p1,NULL,mythread,"A");
14     pthread_join(p1,NULL);
15     pthread_create(&p2,NULL,mythread,"B");
16     pthread_join(p2,NULL);
17     pthread_create(&p3,NULL,mythread,"C");
18     pthread_join(p3,NULL);
19
20     printf("Final Balance is %d\n",balance);
21 }
```



- (A) Assuming none of the system calls fail, when thread p1 prints "Balance is %d", what will p1 say is the value of balance?
- (B) Assuming none of the system calls fail, when the main parent thread prints "Final Balance is %d", what will the parent thread say is the value of balance?

- 5 8. Consider a clinic with one doctor and a very large waiting room (of infinite capacity). Any patient entering the clinic will wait in the waiting room until the doctor is free to see her. Similarly, the doctor also waits for a patient to arrive to treat. All communication between the patients and the doctor happens via a shared memory buffer. Any of the several patient processes, or the doctor process can write to it. Once the patient "enters the doctors office", she conveys her symptoms to the doctor using a call to `consultDoctor()`, which updates the shared memory with the patient's symptoms. The doctor then calls `treatPatient()` to access the buffer and update it with details of the treatment. Finally, the patient process must call `noteTreatment()` to see the updated treatment details in the shared buffer, before leaving the doctor's office. A template code for the patient and doctor processes is shown below. Enhance this code to correctly synchronize between the patient and the doctor processes. Your code should ensure that no race conditions occur due to several patients overwriting the shared buffer concurrently. Similarly, you must ensure that the doctor accesses the buffer only when there is valid new patient information in it, and the patient sees the treatment only after the doctor has written it to the buffer. **You must use only semaphores to solve this problem.** Clearly list the semaphore variables you use and their initial values first. Please pick sensible names for your variables.

(a) Semaphore variables and initial values:

(b) Patient process:

(c) Doctor process:

`consultDoctor();`

`while(1){`

`noteTreatment();`

`treatPatient();`

`}`