Name : Anvit Gupta Enroll : 22114009 Batch : CSE-01

23/02/2024

Indian Institute of Technology Roorkee

Mid-Term Examination (CLOSED BOOK)

Course: Operating Systems (CSN-232)

Duration: 1.5 hr. Max. Marks: 25

Q1. A single processor system has three resource types X, Y and Z, which are shared by three processes. There are 5 units of each resource type. Consider the following scenario, where the column **allocated** denotes the number of units of each resource type allocated to each process, and the column request denotes the number of units of each resource type requested by a process in order to complete execution.

	Allocated			**Request**		
	X	Y	Z	X	Y	Z
P0	1	2	1	1	0	3
P1	2	0	1	0	1	2
P2	2	2	1	1	2	0

What is the order of processes to be finished their execution so that the system is in safe state?

Q2. A system has 4 processes (A, B, C, D) and 5 resources. The current allocation and maximum needs are as follows-

	Allocated					Maximum				
A	1	0	2	1	1	1	1	2	1	3
B	2	0	1	1	0	2	2	2	1	0
C	1	1	0	1	1	2	1	3	1	1
D	1	1	1	1	0	1	1	2	2	0

If Available = [0 0 X 1 1], what is the smallest value of x for which this is a safe state?

Q3. Consider the following set of processes that need to be scheduled on a single CPU. All the times are given in milliseconds. Using the shortest remaining time first scheduling algorithm, what is the average process turnaround time (in milliseconds)?

Process ID	Arrival Time	Execution Time
P1	0	6
P2	3	2
P3	5	4
P4	7	6
P5	10	3

Q4. Consider a system with four processes that arrive in the following order (times are given in milliseconds): (pre-emptive)

Process	Arrival Time	Burst Time	Priority
P1	0	6	3
P2	2	4	1
P3	4	2	4
P4	6	5	2

Calculate the average waiting time and throughput for the system using ***priority scheduling***. Assume that lower priority numbers indicate lower priority (i.e., priority 1 is the lowest priority). Also draw the Gantt chart.

Q5. Consider the following code snippet using semaphores:

```
sem_t mutex = 1;
sem_t s = 0;

void *thread1(void *arg) {
    while (1) {
        sem_wait(&mutex);
        // Critical Section
        sem_post(&s);
        sem_post(&mutex);
        sleep(1);
    }
}

void *thread2(void *arg) {
    while (1) {
        sem_wait(&s);
        // Critical Section
        sem_post(&mutex);
        sleep(1);
    }
}
```

A. Explain the purpose of the semaphore ***s*** in the given code.
B. Discuss the potential problem(s) that may arise from the use of this semaphore and suggest a modification to the code to address such issues.

Q6. We want to use semaphores to implement a shared critical section (CS) among three threads T1, T2, and T3. We want to enforce the execution in the CS in this order: First T2 must execute in the CS. When it finishes, T1 will then be allowed to enter the CS; and when it finishes T3 will then be allowed to enter the CS; when T3 finishes then T2 will be allowed to enter the CS, and so on, (T2, T1, T3, T2, T1, T3….)

What is the minimum number of binary semaphores we need in order to enforce this ordering? Justify your answer.

Q7. Let m[0]...m[4] be mutexes (binary semaphores) and P[0] .... P[4] be processes. Suppose each process P[i] executes the following:

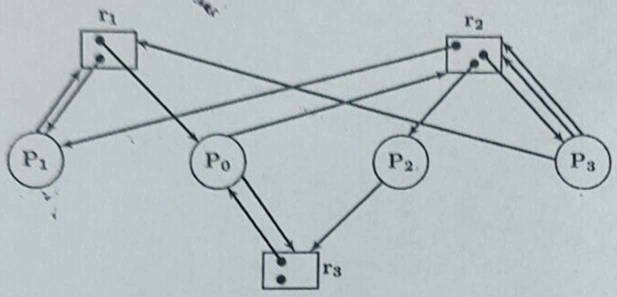
wait (m[i]); wait(m[(i+1) mode 4]);

------

release (m[i]); release (m[(i+1)mod 4]);

What potential problem(s) may arise due to the use of such synchronization mechanism? Justify your answer.

Q8. Find if the system is in a deadlock state or not. Justify your answer by proving the deadlock or identifying the safe sequence.



Q9. A process executes the following code

```
for (i = 0; i < n; i++)
    fork();
```

Write the total number of child processes created as a function of *n*.

Q10. Calculate the number of times "*hello*" will be printed:

```
int main()
{
    fork();
    fork();
    fork();
    printf("hello\n");
    return 0;
}
```

--- END ---