SYSTEMS & DEVICES 2 (SYS2 - COM00021I)
FORMATIVE EXERCISE 1

LECTURER - DR POONAM YADAV

Covering Operating Process Scheduling, Synchronisation and Deadlocks.

The questions for the assignment exercises are adapted from the book - Operating System Concepts by Abraham Silberschatz (author), Peter B. Galvin (author), Greg Gagne (author), 10th edition.

Please try to complete it by 18th Nov 2021 and ask doubts on 22nd Nov 2021 drop-in session for discussions (see your time-table and VLE for more details).

(1) A CPU scheduling algorithm determines an order for the execution of its scheduled processes. Given n processes to be scheduled on one processor, how many possible different schedules are there? Give a formula in terms of n.
(A) $n(n-1)$
(B) n
(C) n^2
(D) $n!$
(E) $n/2$

(2) Consider the following set of processes, with the length of the CPU burst given in milliseconds:

Process	Burst Time	Priority
P1	5	4
P2	3	1
P3	1	2
P4	7	3
P5	4	2

The processes are assumed to have arrived in the order P1, P2, P3, P4, P5, all at time 0.
(a). Draw four Gantt charts that illustrate the execution of these processes using the following scheduling algorithms: FCFS, SJF, nonpreemptive priority (a larger priority number implies a higher priority), and RR (quantum = 2).
(b). What is the turnaround time of each process for each of the scheduling algorithms in part a?
(c). What is the waiting time of each process for each of these scheduling algorithms?

(d). Which of the algorithms results in the minimum average waiting time (over all processes)?

(3) Which of the following scheduling algorithms could result in starvation?
   (a). First-come, first-served
   (b). Shortest job first
   (c). Round robin
   (d). Priority

(4) Consider a system running ten I/O-bound tasks and one CPU-bound task. Assume that the I/O-bound tasks issue an I/O operation once for every millisecond of CPU computing and that each I/O operation takes 10 milliseconds to complete. Also assume that the context-switching overhead is 0.1 millisecond and that all processes are long-running tasks. Describe the CPU utilization for a round-robin scheduler when:
   (a). The time quantum is 1 millisecond
   (b). The time quantum is 10 milliseconds

(5) Explain why spinlocks are not appropriate for single-processor systems yet are often used in multiprocessor systems.

(6) Race conditions are possible in many computer systems. Consider an online auction system where the current highest bid for each item must be maintained. A person who wishes to bid on an item calls the bid(amount) function, which compares the amount being bid to the current highest bid. If the amount exceeds the current highest bid, the highest bid is set to the new amount. This is illustrated below:

```
void bid(double amount) {
if (amount > highestBid)
highestBid = amount;
}
```

Describe how a race condition is possible in this situation and what might be done to prevent the race condition from occurring.

(7) A multithreaded web server wishes to keep track of the number of requests it services (known as hits). Consider the two following strategies to prevent a race condition on the variable hits. The first strategy is to use a basic mutex lock when updating hits:

```
int hits;
mutex lock hit lock;

hit lock.acquire();
hits++;
hit lock.release();
```

A second strategy is to use an atomic integer:

```
atomic_t hits;
atomic_inc(&hits);
```

Explain which of these two strategies is more efficient.

(8) Discuss the tradeoff between fairness and throughput of operations in the readers–writers problem. Propose a method for solving the readers–writers problem without causing starvation.

(9) Consider the following snapshot of a system:

	Allocation	Max	Available
	A B C D	A B C D	A B C D
T_0	3 1 4 1	6 4 7 3	2 2 2 4
T_1	2 1 0 2	4 2 3 2	
T_2	2 4 1 3	2 5 3 3	
T_3	4 1 1 0	6 3 3 2	
T_4	2 2 2 1	5 6 7 5	

Answer the following questions using the banker's algorithm:

(a). Illustrate that the system is in a safe state by demonstrating an order in which the threads may complete.

(b). If a request from thread T4 arrives for (2, 2, 2, 4), can the request be granted immediately?

(c). If a request from thread T2 arrives for (0, 1, 1, 0), can the request be granted immediately?

(d). If a request from thread T3 arrives for (2, 2, 1, 2), can the request be granted immediately?