No of Pages : 4 **Course Code : 15XW44 / 15XT44 /15XD44**

Roll No:
(To be filled in by the candidate)

PSG COLLEGE OF TECHNOLOGY, COIMBATORE - 641 004

SEMESTER EXAMINATIONS, DECEMBER 2017

MSc – TCS / SOFTWARE SYS.,/DATA SCIENCE Semester : 4

15XW44 / 15XT44 / 15XD44 OPERATING SYSTEMS

Time : 3 Hours **Maximum Marks : 100**

INSTRUCTIONS:
1. Answer **ALL** questions. Each question carries 20 Marks.
2. Subdivision (a) carries 3 marks each, subdivision (b) carries 7 marks each and subdivision (c) carries 10 marks each.

1. a) Application S is implemented with 6 threads and it executes in a 4-core system. Thread s1 reads data from network, threads s2-s5 process the data, and thread s6 will save processed summary data into file system. Would it be advisable to implement S as kernel level, or as user level threads? Why?

b) Define the following types of operating systems

(i) Batch (ii) Multiprogramming (iii) Timesharing
(iv) Real time (v) Distributed

c) Draw the process state transition diagram of an OS in which i) each process is in one of the five states: created, ready, running, blocked (i.e. sleep or wait), or terminated, and ii) only non-preemptive scheduling is used by the OS. Explain the state transitions.

The table below is a representation of the OS's internal process table. Each process has a stack, PID, a status, a priority and a next field. The next field is used to store the index of the process table using two circular linked list. One list contains all the ready and running processes (0 -> 4 > 3 -> 2 ->0) and another list contains all the blocked processes (1->1). Assume that a higher priority value means a higher priority.

	Stack	PID	Status	Priority	Next	CPU Burst
0	Ptr	4	Ready	1	4	33
1	Ptr	5	Blocked	10	1	134
2	Ptr	7	Running	7	0	58
3	Ptr	11	Ready	5	2	150
4	Ptr	13	Ready	8	3	145

- Suppose that round-robin scheduling is being used. Which process would be run after the current time slice expires?
- Suppose that priority scheduling is being used and process 7 makes a blocking request. Which process would be run next?
- Suppose that process 7 makes a blocking request and round robin scheduling is being used. What would the table look like?
- Suppose that while process 7 is running, process 5's blocking request is satisfied and it is no longer blocking. What would the process table look like (just before the next process context switch)?
- Suppose a new process with PID 24 and priority 6 is created. What would the process table look like?

2. a) What would be good size for time quantum in round robin (time slice) scheduling? When would time quantum be too small, and when too large?

b) A single-lane bridge connects the two Vermont villages of North Tunbridge and South

Tunbridge. Farmers in the two villages usethis bridge to deliver their produce to the neighboring town. The bridge can become deadlocked if a northbound and a southbound farmer get on the bridge at the same time. (Vermont farmers are stubborn and are unable to back up) Using semaphores and/or mutex locks, design an algorithm in pseudocode that prevents deadlock.

- Using exactly one semaphore, design an algorithm that prevents deadlock. Initially, do not be concerned about starvation (the situation in which northbound farmers prevent southbound farmers from using the bridge, or vice versa).
- Justify that your solution is starvation-free.

c) Consider a set of 7 processes named P1 through P7. Arrival time, the execution time of each process and the corresponding queue in which they are placed are as given below. Scheduling among queues is based on fixed priority with 1st Foreground has higher priority and background having lowest priority. By "**Multilevel Queue**" scheduling algorithm, draw the CPU scheduling Gantt chart and compute the average waiting and turnaround time. Assume Context Switch Time is 0.

Queue	Process	Burst time	Arriving time	Algorithm
■ 1st Foreground	P1	50	0.0	RR(TQ:20)
	P2	15	30.0	
	P3	45	30.0	
■ 2nd Foreground	P4	40	0.0	SJF Preemptive
	P5	10	120.0	
■ Background	P6	30	60.0	FCFS
	P7	20	130.0	

3. a) What are the major differences between deadlock, starvation, and race? Comment on the following statements.
   i) A starvation-free solution is also deadlock-free
   ii) A deadlock free solution is starvation free

   b) In a fixed-partitioning scheme, state two advantages of using unequal-size partitions? Given five memory partitions of 100 KB, 500 KB, 200 KB, 300 KB, and 600 KB (in order), how would each of the first-fit, best-fit, and worst-fit algorithms place processes of 212 KB, 417 KB, 112 KB, and 426 KB (in order)?Which algorithm makes the most efficient use of memory?

   c) **Explain what deadlock avoidance is and apply it** to the following problem. A restaurant would like to serve four dinner parties, P1 through P4. The restaurant has a total of 8 plates and 12 bowls. Assume that each group of diners will stop eating and wait for the waiter to bring a requested item (plate or bowl) to the table when it is required. Assume that the diners don't mind waiting. The maximum request and current allocation tables are shown as follows:

Maximum Request	Plates	Bowls
P1	7	7
P2	6	10
P3	1	2
P4	2	4

Current Allocation	Plates	Bowls
P1	2	3
P2	3	5
P3	0	1
P4	1	2

- Determine the Need Matrix for plates and bowls.

Need	Plates	Bowls
P1		
P2		
P3		
P4		

- Will the restaurant be able to feed all four parties successfully? Clearly explain your answer – specifically, why no or why/how there is a safe serving order.
- Assume a new dinner party, P5, comes to the restaurant at this time. Their maximum needs are 5 plates and 3 bowls. Initially, the waiter brings 2 plates to them. In order to be able to feed all five parties successfully, the restaurant needs more plates.
  - Determine the new Need Matrix for plates and bowls.

Need	Plates	Bowls
P1		
P2		
P3		
P4		
P5		

  - At least how many plates would the restaurant need to add?
  - Show a safe serving sequence.

4. a) If a large number of programs is kept in main memory, then there is almost always another ready program when a page fault occurs. Thus, CPU utilization is kept high. If, however, we allocate a large memory space toeach of a few programs, then each program produces a smaller number ofpage faults. Thus, CPU utilization is kept high. Are these two argumentscorrect? Which policy, if either, should be preferred? Why?

b) When do page faults occur? List the actions taken by the operating system when a page fault occurs. Consider the following sequence of memory references from a 460 bytes size program:

10, 11, 104, 170, 73, 309, 185, 245, 246, 434, 458, 364.

- Give the reference string, assuming a page size of 100 bytes.
- Find the page-fault rate for the reference string in part (i), assuming 200 bytes (2 frames) of main memory is available to the program and a FIFO replacement algorithm is used.
- Calculate the page fault rate if the page replacement algorithm is LRU.

c) When using physical addresses directly, there is no virtual to physical translation overhead. Assume it takes 200 nanoseconds to make a memory reference. If we used physical addresses directly, then all memory referenceswill take 200 nanoseconds each.

- If we use virtual addresses with page tables to do the translation, then without a TLB we must first access the page table to get the appropriate page table entry (PTE) for translating an address, do the translation, and then make a memory reference. Assume it also takes 200 nanoseconds to access the page table and do the translation. In this scheme, what is the effective memory reference time (time to access the page table + time to make the memory reference)?
- If we use a TLB, PTEs will be cached so that translation can happen as part of referencing memory. But, TLBs are very limited in size and cannot hold all PTEs, so not all memory references will hit in the TLB. Assume translation using the

TLB adds no extra time and the TBL hit rate is 75%. What is the effective average memory reference time with this TLB?

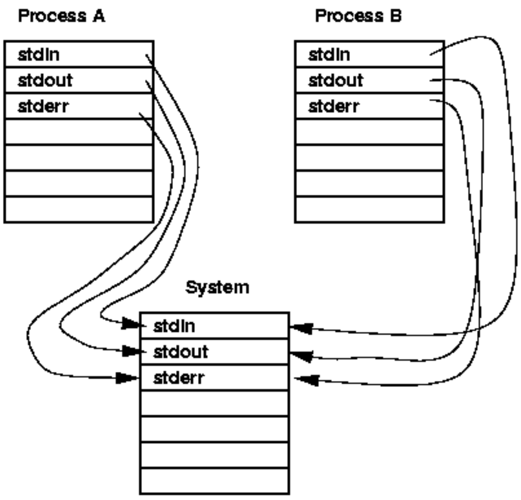
- If we use a TLB that has a 99.5% hit rate, what is the effective average memory reference time now? (This hit rate is close to what TLBs typically achieve in practice.)

5. a) Give an example situation where indexed sequential file would be best file organization type. Why would Sequential File be not good with your example? Why would Indexed File be not good with your example?

b) Disk requests come into the disk driver for cylinders 10, 22, 20, 2, 40, 6, and 38, in that order. Assume that the disk has 100 cylinders.
A seek takes 6msec per cylinder moved. Compute the average seek time for the request sequence given above for
   - First-come, First-served
   - Shortest Seek Time First (SSTF)
   - LOOK (with the disk-arm initially moving towards higher number cylinders from lower number cylinders)

In all the cases, the arm is initially at cylinder 20.

c) What file system structures are maintained: (i) on disk, (ii) in memory? Explain how these data structures are used during file open and file read operations. Consider two independent processes A and B that have just started executing along with per-process and system open file tables with the following initial contents:



- Each of the following pieces of data must be kept in either the per-process open file table or the system open file table. Determine which of the two locations is most appropriate for each piece of information. Justify your answer.
   - Mode (e.g., RDONLY, WRONLY, or RDWR) used to open the file
   - Open file reference count
   - Meta-data (also kept on disk) such as file access and modification time, file size, and pointers to data blocks
   - Current read or write position in file
- Modify the diagram to show the final state of all tables after the following system calls have executed. Show the names of the file assigned to each table entry and the pointers from the per-process table to the system table. Include the **reference count** in the appropriate table(s). Assume that process A and B have the same current working directory and that all system calls complete successfully.
Process B: fd1 = open("x", mode);
Process A: fd1 = open("y", mode);
Process B: fd2 = open("y", mode);
Process A: close(stdin);

Process A: fd3 = open("z", mode);

/END/

FD/JU