Sample Questions

Amir H. Payberah
amir@sics.se

Amirkabir University of Technology
(Tehran Polytechnic)

Question 1

- Suppose a thread is running in a critical section of code, meaning that it has acquired all the locks through proper arbitration. Can it get context switched? Why or why not?

Answer 1

- Yes, it can get context switched. Locks (especially user-level locks) are independent of the scheduler.

Question 2

- What is a thread? What is a process? Describe how to create each of these.

Answer 2

- A thread is a piece of an executing program. It contains a program counter (PC), processor registers and a stack. A process consists of one or more threads in an address space. Threads are created by allocating a new stack and Thread Control Block (TCB), initializing the registers in the TCB properly, then placing the new TCB on the ready queue.
- A process is created by allocating a new Process Control Block (PCB), address space descriptor/page tables, and a new thread. In UNIX, processes are created by a `fork()` system call which also creates a complete copy of the parent process for the new (child) process.

Question 3

- What is a thread-join operation?

Answer 3

- Thread-join is issued by one thread when it wants to stop and wait for the termination of another thread. On termination of the target thread, the original thread resumes execution.

Question 4

- What is an interrupt? What happens when an interrupt occurs? What is the function of an interrupt controller?

Answer 4

- An interrupt is an external signal. When interrupt occurs and is enabled, then the processor will stop the current running code, save the Program Counter (PC), disable interrupts, and jump to an appropriate interrupt handler.
- The function of the interrupt controller is to provide some control over which hardware interrupt signals are enabled and what their priority is.

Question 5

- What is a monitor?

Answer 5

- A monitor is a synchronization mechanism that abstracts away protection and scheduling access to shared data through the use of a lock and zero or more condition variables.

Question 6

- Six jobs are waiting to be run. Their expected running times are 10, 8, 6, 3, 1, and X. In what order should they be run to minimize average completion time? State the scheduling algorithm that should be used and the order in which the jobs should be run. (Your answer will depend on X).

Answer 6

- Shortest Job First or Shortest Remaining Processing Time
- $X \leq 1$: X, 1, 3, 6, 8, 10
- $1 < X \leq 3$: 1, X, 3, 6, 8, 10
- $3 < X \leq 6$: 1, 3, X, 6, 8, 10
- $6 < X \leq 8$: 1, 3, 6, X, 8, 10
- $8 < X \leq 10$: 1, 3, 6, 8, X, 10
- $X > 10$: 1, 3, 6, 8, 10, X

Question 7

- Name and describe 4 different scheduling algorithms. What are the advantages and disadvantages of each?

Answer 7 (1/4)

- FCFS (FIFO): The first process that arrives gets to run all the way to completion
  - + Simple and quick to choose the next thread
  - + Fair in the sense that jobs run in the order they arrived
  - - Long wait times for short jobs that get stuck behind long jobs
  - - Not fair for jobs that can finish very quickly but must wait a long tim

Answer 7 (2/4)

- Round Robin: Scheduler rotates through all threads, allowing each to run for some time quantum
  - + Fair because gives every thread a chance to run
  - - Potentially large overhead from frequent context switching
  - - Can result in large completion times, especially for threads with similar run times

Answer 7 (3/4)

- Shortest Job First: Scheduler runs the shortest job first, non-preemptive
  - + Results in low average completion time
  - - Long jobs can et stuck behind short jobs
  - - Requires knowledge of the future, hard to predic

Answer 7 (4/4)

- Shortest Remaining Processing Time: Works like SJF, but preemptive
  - +/- Same as SJF

Question 8

- Here is a table of processes and their associated running times. All of the processes arrive in numerical order at time 0. Show the scheduling order for these processes under 3 policies: First Come First Serve (FCFS), Shortest-Remaining-Time-First (SRTF), Round-Robin (RR) with timeslice quantum $= 1$.

Process ID	CPU Running Time
1	2
2	6
3	1
4	4
5	3

Answer 8

Time slot	FCFS	SRTF	RR
0	1	3	1
1	1	1	2
2	2	1	3
3	2	5	4
4	2	5	5
5	2	5	1
6	2	4	2
7	2	4	4
8	3	4	5
9	4	4	2
10	4	2	4
11	4	2	5
12	4	2	2
13	5	2	4
14	5	2	2
15	5	2	2

Question 9

- For each process in each schedule above, indicate the queue wait time and completion time (otherwise known as turnaround time, TRT). Note that wait time is the total time spend waiting in queue (all the time in which the task is not running):

Answer 9

Scheduler	P1	P2	P3	P4	P5
FCFS wait	0	2	8	9	13
FCFS TRT	2	8	9	13	16
SRTF wait	1	10	0	6	3
SRTF TRT	3	16	1	10	6
RR wait	4	10	2	10	9
RR TRT	6	16	3	14	12

Question 10

- What are three methods of dealing with the possibility of deadlock? Explain each method and its consequences to programmer of applications.

Answer 10

- Ignore it: do nothing about it. Application programmers have to deal with deadlock or not care if it happens
- Prevent it: use Bankers Algorithm and only allow allocations that cannot possibly lead to deadlock. Application programmer has to specify maximum possible resource needs
- Detect it and recover: check dependency graph for cycles and kill a thread in the cycle. Application programmer has to deal with their threads possibly getting killed.

Question 11

- Suppose that we have several processes and several resources. Name three ways of preventing deadlock between these processes.

Answer 11

- Atomically acquire all resources at the start of execution
- Use the Banker's Algorithm
- Acquire all resources in a specified order

Question 12

- Suppose that we have the following resources: A (12 instances), B (9 instances), C (12 instances) and four threads T1, T2, T3, T4. Further, assume that the processes have the following maximum requirements and current allocations. Is the system potentially deadlocked (i.e., unsafe)? In making this conclusion, you must show all your steps, intermediate matrices, etc.

Maximum

Thread ID	A	B	C
T1	4	3	4
T2	5	3	3
T3	6	4	3
T4	4	1	2

Current Allocation

Thread ID	A	B	C
T1	2	1	3
T2	1	2	3
T3	5	4	3
T4	2	1	2

Answer 12

Available

A	B	C
2	1	1

Need

Thread ID	A	B	C
T1	2	2	1
T2	4	1	0
T3	1	0	0
T4	2	0	0

Available after running

Thread ID	A	B	C
T4	4	2	3
T3	9	6	6
T2	10	8	9
T1	12	9	12

- No. Running the Bankers Algorithm, we see that we can give T4 all the resources it might need and run it to completion.

Questions?