# Section 10: Scheduling

## Question 1

1. Describe the difference between an IO-Bound and a Compute-Bound process.
2. Describe Processor Burst Time.
3. What differences between IO and Compute bound process’s effect on Processor Burst Time?

### Answer

1. IO-Bound: Processes that make frequent I/O requests or system calls that cause the process to block e.g. an application that is retrieving and persisting data to files such as a database.
2. Compute-Bound: Processes that spend most of their time processing data and will run for extended periods without making blocking system calls.
3. Processor burst time is the time a process spends executing before being blocked (I/O) or preempted (timer interrupt).
4. A process that is compute-bound will have longer processor burst times. Compute bound processes will not often block for I/O and so are more likely to run for their entire scheduling quantum.

## Question 2

Consider Throughput vs Response Time as scheduling criteria.

1. Provide a definition of System’s Response Time.
2. How can the OS minimize an interactive process’s response time?
3. Is minimizing response time critical to a batch processing system? If not, which scheduling criteria is important to batch processing systems?

### Answer

1. The system’s response time is perceived by the user as the time between making a request of the system (i.e. mouse click, text entry, etc.) and the system’s response to the request.
2. Response time can be minimized by keeping the pages for an interactive processes resident in memory (not swapping out those pages) even if the pages are not currently active (in the working set).
3. Minimizing response time is not critical to a batch processing system. Maximizing system throughput is important to a batch processing system.

## Question 3

1. Describe *Tr  and Ts* , and the meaning of the ratio *Tr /Ts* .
2. Describe how Non-Preemptive FIFO dispatching can produce an unfair schedule when there is a mix of long and short-lived processes.

### Answer

1. Tr /Ts is ratio of the Turnaround Time / Service Time. This ratio was described in class as the ‘Fairness Ratio”.  
     
   The fairness ratio describes the delay the process experienced before completing. The minimum ratio is 1 meaning that the process executed to completion as soon as it was submitted for processing. The ratio grows if the process’s execution is delayed. In our examples, the delay is because of unfair scheduling policies.
2. With N-P FIFO, if a short-lived process is submitted behind one or more long-lived compute-bound processes, the execution of the short-lived process is delayed until the processes ‘in front of it’ have completed, increasing its Tr. This might be considered an unfair schedule for the short lived process and can lead to user dissatisfaction when the expectation is that short running processes should finish quickly.

## Question 4

Describe the problem with fairness with Preemptive Round-Robin scheduling WRT I/O-bound processes vs. compute-bound processes.

### Answer

RR scheduling allows a process to complete its quantum (time-slice) and then schedules the next process at the head of the queue. A compute-bound process will likely execute for its full quantum without blocking. However, an I/O-bound process is more likely not to use its entire quantum before being blocked (I/O) and rescheduled when the I/O operation completes. Thus overall, an I/O-bound process will receive less processor time than a compute-bound process.

## Question 5

How does Virtual Round-Robin scheduling (as described in text) solve the problem described in question 5?

### Answer

Virtual RR scheduling maintains a second queue for “Returned From Blocked” processes that have completed their I/O operation and have not completed their current quantum. See Figure 9.7. When a process is placed in the RFB queue, the dispatcher will preempt the currently executing process in favor of the returned-from-blocked process which will be allowed to complete the remainder of its previously interrupted quantum before being placed into the ready queue.

## Question 6

1. Briefly describe how the dispatcher selects the next ready process to execute when scheduling processes of different priorities.
2. How does the slide suggest implementing a priority-aware dispatcher? Hint: dispatch queues.

### Answer

1. The dispatcher will give preference to (selects) processes of higher priority over processes of lower priority.
2. Suppose each process is assigned a priority of 1-N, where N is higher priority than N-1.  
   The dispatcher can be implemented with N FIFO queues where each queue holds ready processes of priority i. The dispatcher will select from the highest priority, non-empty queue. For example, the dispatcher will select a process from Queue Lvl N-1 only if Lvl N is empty.

## Question 7

1. How does Feedback scheduling give preference to short lived processes over long lived processes?
2. How does Feedback scheduling keep long-lived processes from starving?

### Answer

1. Feedback scheduling assigns priorities to processes according to the number of times they have timed out (completes its quantum). The more times a processes times-out, the lower its assigned priority becomes. This will cause new processes to be scheduled ahead of long-lived processes.
2. The scheduling quantum assigned to a process is increased as its priority is decreased. This means that although a long-lived process is not scheduled as often, when it is scheduled it is allowed to execute for a longer duration before timing out.

## Question 8

What is the tradeoff between long and short scheduling quantum durations?

### Answer

Long quantum durations will unfairly favor the execution of compute-bound processes as they will receive a larger percentage of the processor before being timed out. Recall that IO-bound processes run for short times and block.

Short quantum durations more fairly schedule all type of processes (including short and I/O bound), but decreases processor utilization with an increase in the OS overhead caused by increased context switching.

## Question 9

1. In the context of Multiprocessor Scheduling, describe the difference between Static Assignment vs. Dynamic Assignment of threads to processors.
2. As described in the slides, how are both strategies implemented? Hint: ready queues.

### Answer

1. With static assignment, newly created threads are attached to, and execute on, the same processor throughout is lifetime (unless moved by a load balancer).   
   With dynamic assignment, ready threads are assigned by the dispatcher to any available processor i.e. threads migrate between processors.
2. Static assignment is implemented with per-processor ready queues.   
   Dynamic assignment is implemented with a single, global ready queue.

## Question 10

In terms of processor cache utilization, what is the advantage of repeatedly dispatching (pinning) threads to the same processor?

### Answer

As a thread executes, its current locality causes certain text and data pages to be loaded into its processor’s cache (L1 & L2). If the thread is re-scheduled to the same processor, the processor’s cache remains ‘hot’ i.e. all of the thread’s pages will remain in cache. Contrast with the situation where a thread executes on a different processor; each thread starts its execution with a cold cache that must be reloaded with the thread’s text and data pages.

## Question 11

In terms of throughput, what is the advantage of Gang Scheduling?

### Answer

Gang scheduling attempts to keep all of a process’s threads executing simultaneously. This is important when a process’s threads are tightly coupled, as they often are. Tightly coupled threads will block when synchronizing with a non-running peer thread. The blocking results in a type of ‘processor thrashing’ where a thread executes for a short period and blocks waiting to synchronize with a peer. When all of the threads are running simultaneously, there is no synchronization blocking, the process completes sooner.

## Question 12

1. Describe the meaning of Load Balancing in the context of Processor Assignment.
2. Is a Load Balancer needed with dynamic or static processor assignment?

### Answer

1. Load balancing is the assignment of threads to processors with the goal of spreading the assignment of threads evenly across all processors. The ‘Load Balancer’ is a task that periodically runs and examines the load each processor is under. A processor’s load may be measured in terms of the number of threads assigned to the processor’s queue (static assignment).
2. Load balancing is only needed with static assignment. Dynamic assignment naturally spreads the load as threads are assigned from a single global queue to the next available (idle) processor.

## Question 13

Describe the meaning of, and relationships between, events, tasks, and deadlines in the context of real-time operating systems.

### Answer

An Event is a signal (usually an interrupt) that triggers the execution of a task.

An Event Handling Task is code that is executed (triggered) in response to the event. A Periodic Task is periodically scheduled for execution by the RT OS.

A Deadline is a measurement of time between when an event occurs and the completion of the execution of a task that responds to the event and generates an output. The system must respond to, and produce the output within a specific amount of time (its deadline) or suffer some type of failure.

From the example of automotive airbag controllers given in class, a collision (**event**) is detected by an accelerometer and signals the automobile’s on-board computer. The system executes a specific **task** that examines the input and decides whether the air bags should be deployed. The system must respond to the ‘crash’ event and cause the airbags to deploy within a certain amount of time (**deadline**) otherwise the airbag effectiveness will be reduced or even become a hazard to the passengers.

## Question 14

What are the characteristics of how real-time operating systems manage and execute tasks within their deadlines?

### Answer

1. Execution of the task is triggered by events (interrupt), not the OS scheduler.
2. Memory hosting the task’s code and data segments cannot be swapped out to disk i.e. no virtual memory.
3. Tasks cannot make any blocking system calls i.e. calls to IO or network communication.
4. The task must be designed and verified to execute well within the deadline imposed by the application and the processing capacity of the processor. Worse-case path analysis must be employed to ensure that no possible combination of state and inputs can cause the task complete after its deadline.

## Question 15

Describe the two types of tasks dispatched by real-time operating systems?

### Answer

1. Event-Handling Tasks that are executed in response to events from external sources.
2. Periodic Tasks that are periodically executed (scheduled) by the operating system.