# CSI 3131 Midterm Sample – W7

# CSI 3131 Midterm Sample – W15

# CSI 3131 Midterm Sample – S18

1. **To access the services of operating system, the interface is provided by:**
   1. System Calls
   2. API
   3. Library
   4. Assembly Instructions
   5. None of the above
2. **Which one of the following is not true?**
   1. Kernel is the program that constitutes the central core of the operating system
   2. Kernel is the first part of the operating system to load into memory during booting
   3. Kernel is made of various modules which will be loaded on demand
   4. Kernel remains in the memory during the entire computer session
3. **In operating systems, each process has its own:**
   1. Address space and global variables
   2. Open files
   3. Pending alarms, signals and signal handlers
   4. All of the above mentioned
   5. None of the above mentioned
4. Which system call returns the process identifier of a terminated child?
   1. wait
   2. exit
   3. fork
   4. get
   5. exec
5. The number of processes completed per unit time is known as \_\_\_.
   1. Response Time
   2. Throughput
   3. Efficiency
   4. Turnaround Time
   5. None of the above
6. The Process Control Block is:
   1. Data Structure
   2. A secondary storage section
   3. A Block in memory
   4. All of the above
   5. None of the above
7. The degree of multi-programming is:
   1. The number of processes executed per unit time
   2. The number of processes in the ready queue
   3. The number of processes in the I/O queue
   4. The number of processes in memory
   5. None of the above
8. The parent process completes execution, but the child keeps executing, then the child process is known as:
   1. Orphan
   2. Zombie
   3. Body
   4. Dead
   5. None of the above
9. In Java, you can create a thread by:
   1. Extending the Object class or using the intrinsic lock
   2. Calling fork system call
   3. Implementing the Runnable interface or extending the Thread class
   4. Creating immutable objects out of the Task class
   5. Extending the Runnable class or implementing the Thread interface
10. **A monitor is a module that encapsulates:**
    1. Shared data structures
    2. Procedures that operate on shared data structure
    3. Synchronization between concurrent procedure invocation
    4. All of the mentioned
    5. None of the above
11. What is a medium-term scheduler?
    1. It selects which process has to be brought into the ready queue
    2. It selects which process has to be executed next and allocates CPU to it
    3. It selects which process to remove from memory by swapping
    4. All of the above
    5. None of the above
12. In Round Robin scheduling, when the time quantum given to a process expires, the process goes from the \_\_\_ state to the \_\_\_ state:
    1. Running, Blocked
    2. Ready, Running
    3. Ready, Suspended
    4. Running, Ready
    5. Blocked, Ready
13. The interval from the time of submission of a process to the time of completion of a process burst is termed as:
    1. Waiting time
    2. Turnaround time
    3. Response time
    4. Real time
    5. Time quantum
14. In priority scheduling algorithm, when a process arrives at the ready queue, its priority is compared with the priority of:
    1. All processes
    2. Blocked processes
    3. Parent process
    4. Child process
    5. Running process
15. Which one of the following cannot be scheduled by the kernel?
    1. Kernel thread
    2. User thread
    3. Process
    4. Interrupt service routine
    5. None of the mentioned
16. An I/O bound program will typically have:
    1. Few very short CPU bursts
    2. Many very short I/O bursts
    3. Many very short CPU bursts
    4. Few very short I/O bursts
    5. Many very long CPU bursts
17. Bounded waiting implies that there exists a bound on the number of times a process is allowed to enter its critical section:
    1. The process is allowed to enter the critical section before other processes can make a request to enter the critical section
    2. Other processes are allowed to enter the critical section before the process can enter the critical section after it made the request to enter the critical section
    3. The process is allowed to enter the critical section before processes can enter the critical section
    4. The process is allowed to request the critical section after other processes enter the critical section
18. Consider the methods used by processes P1 and P2 for accessing their critical sections whenever needed, as given below. The initial values of shared Boolean variables S1 and S2 are randomly assigned.  
      
    In this situation, the following statements describes properties achieved:
    1. Mutual exclusion but not progress
    2. Progress but not mutual exclusion
    3. Neither mutual exclusion nor progress
    4. Both mutual exclusion and progress

# CSI 3131 Midterm Sample – S22

Answer the following multiple-choice questions:

1. **\_\_\_\_ scheduling is approximated by predicting the next CPU burst with an exponential average of the measured lengths of previous CPU bursts.**
   1. **FCFS (First Come First Serve)**

**First Come First Serve (FCFS) schedules processes as they are called.**

* **Simple and easy to program:** YES
* **Other Benefits:** Cost little time to make scheduling decisions
* **Low Waiting Time:** NO – it is very expensive.
* **Good CPU Utilization:** NO – the convoy effect
  + **Convoy Effect:** All the other processes wait for the one big process to get off the CPU. This effect results in lower CPU and device utilization than might be possible if shorter processes we allowed to go first.
* **Better solution:** Execute ready processes with short CPU bursts. Shortest Job First is an example of a more optimal strategy.
  1. **SJF (Shortest Job First)**

**Shortest Job First (SJF)**

* **Advantages:**
  + Designed to ensure low average waiting time as long as burst-length predictions work.
* **Disadvantages:**
  + Difficult estimating the length of CPU bursts in advance
  + Impossible to know next CPU burst. But it can be estimated.
    - Estimated by length of previous CPU bursts using exponential averaging.
      * **Exponential Averaging**: More weight given to more recent bursts, not just straight averaging.
  + Long processes will suffer from ***famine*** or ***starvation*** when there is a constant supply of short processes (which are always prioritized first)
  + Pre-emption is required for timeshared environments.
    - **Pre-emption**: If a new process arrives with CPU burst length less than remaining time of current executing process, pre-empt. This scheme is also known as the **Shortest-Remaining-Time-First (SRTF)** or **Pre-Emptive Shortest Job First**
      * Pre-emptive Shortest Job First is optimal because it gives minimum average waiting time for a given set of processes. (Moving a process with short CPU burst in front of a process with longer CPU burst reduces average waiting time).
    - **Non Pre-emption**: Once CPU given to the process it cannot be pre-empted until completes its CPU burst.
* **Better Solution**:
  + Priority Scheduling

Shortest Job First scheduling predicts the next CPU burst in order to complete the shortest-job-first. This prediction uses the exponential average of the measured lengths of previous CPU bursts.

Exponential Averaging with Shortest Job First (SJF)

**acual length of CPU burst**

relative weight of a recent vs past history

predicted value for the next CPU burst

* 1. **RR (Round Robin)**

**Priority Scheduling**

* **Features:**
  + First Come First Service scheduling with Pre-emption to enable the system to switch between processes.
  + Time quantum is generally between 10 to 100 milliseconds in length
  + The ready queue is treated as a circular queue, allocating the CPU to each process for a time interval of up to 1 time quantum.
* **Advantages:**
  + More important jobs will be scheduled before lesser important jobs
  + Provides support for real-time processing
* **Disadvantages:**
  + Queues could have a O(n) search to determine the highest-priority process.
  + **Starvation / Famine**: Low priority processes may never execute, (similar to how long processes may starve in Shortest Job First, but instead focuses on priority instead of length)
    - Starvation / Famine can be solved by increasing the priority of aging processes.
  + Same Priority Processes use:
    - First Come First Serve
    - First Come First Serve with Pre-Emption (AKA Round Robin)
* **Better Solution:**
  + Multilevel Queue
  1. **Multilevel queue**

**Multilevel Queue**

* **Features:**
  + Priority scheduling combined with round robin (First Come First Service with Pre-Emption)
    - Separated queues for each distinct priority. Priority scheduling simply schedules the process in the highest-priority queue.
    - If multiple processes in the highest-priority queue, then they are executed in the round-robin order.
    - A priority is assigned statically to each process, and a process remains in the same queue for the duration of its runtime.
  + Multilevel queue scheduling can be used to partition processes based on process type
    - **Process Types**:
      * **Chart, bar chart

        Description automatically generated**
      * Foreground = interactive processes
        + Priority is externally defined over background processes
      * Background = batch processes
    - The two types of processes (foreground and background) have different response-time requirements and may have different scheduling needs.
      * As such, each queue may have its own scheduling algorithm
  + **Scheduling Amongst Queue Partitions:**
    - Commonly implemented as fixed-priority pre-emptive scheduling.
      * Example: real-time queue may have absolute priority over the interactive queue.
* **Advantages:**
  + Solves issue with Priority and Round Robin scheduling where all processes may be placed in a single queue, and the scheduler then selects the process with the highest priority to run.
* **Disadvantages:**
  + More overhead for implementation
  1. Multilevel Feedback Queue

1. **The \_\_\_ scheduling algorithm is designed especially for time-sharing systems.**
   1. **FCFS (First Come First Serve)**

The simplest scheduling algorithm, but it can cause short processes to wait for a very long processes.

* 1. **SJF (Shortest Job First)**

Provably optimal since it provides the shortest average waiting time. Implementing SJF scheduling is difficult because predicting the length of the next CPU burst with exponential averaging is difficult.

* 1. **RR (Round Robin)**

Allocates the CPU to each process for a time quantum. If the process does not relinquish the CPU before its time quantum expires, the process is pre-empted, and another process is scheduled to run for a time quantum.

* 1. **Multilevel queue**

Assigns each process a priority, and the CPU is allocated to the process with the highest priority. Processes with the same priority can be scheduled in First Come First Serve order or using Round Robin scheduling.

1. **How many processes are created at the end of the following loop?**  
   for(int i = 0; i < 3; i++){ fork(); }
   1. 3
   2. 7
   3. 8
   4. 9

2

0

1

1. **What effect does the size of the time quantum have on the performance of an RR algorithm?**

The Round Robin (RR) scheduling algorithm is similar to FCFS scheduling, but pre-emption is added to enable the system to switch between processes. A small unit of time, called a time quantum, is defined. A time quantum is generally from 10 to 100 milliseconds in length. The ready queue is treated as a circular queue. The CPU scheduler goes around the ready queue, allocating the CPU to each process for a time interval of up to 1 time quantum.

To implement Round Robin scheduling, we again treat the ready queue as a First In First Out (FIFO) queue of processes. New processes are added to the tail of the ready queue. The CPU scheduler picks the first process from the ready queue, sets a timer to interrupt after 1 time quantum, and dispatches the processes.

One of two things will then happen. The process may have a CPU burst of less than 1 time quantum. In this case, the process itself will release the CPU voluntarily. The scheduler will then proceed to the next process in the ready queue. If the CPU burst of the currently running process is longer than 1 time quantum, the timer will go off and will cause an interrupt to the operating system. A context switch will be executed, and the process will be put at the tail of the ready queue. The CPU scheduler will then select the next process in the ready queue.

The average waiting time under the RR policy is often long. Consider the following set of processes that arrive at time 0, with the length of the CPU burst given in milliseconds:

|  |  |
| --- | --- |
| **Processes** | **Burst Time** |
| P1 | 24 |
| P2 | 3 |
| P3 | 3 |

If we use a time quantum of 4 milliseconds, then process P1 gets the first 4 milliseconds. Since it requires another 20 milliseconds, it is pre-empted after the first time quantum, and the CPU is given to the next process in the queue, process P2. Process P2 does not need 4 milliseconds, so it quits before its time quantum expires. The CPU is then given to the next process, process P3. Once each process has received 1 time quantum, the CPU is returned to process P1 for an additional time quantum. The resulting RR schedule is as follows:

A picture containing timeline

Description automatically generated

Let’s calculate the average waiting time for this schedule.

P1 waits for 6 milliseconds (10-4=6)

P2 waits for 4 milliseconds (4-0=4)

P3 waits for 7 milliseconds (7-0=7

average waiting time is

In the RR scheduling algorithm, no process is allocated the CPU for more than 1 time quantum in a row (unless it is the only runnable process). If a process’s CPU burst exceeds 1 time quantum, that process is pre-empted and is put back in the ready queue. The RR scheduling algorithm is thus pre-emptive.

If there are processes in the ready queue, and the time quantum is , then each process gets of the CPU time in chunks of at most time units. Each process must wait no longer than time units until its next time quantum. For example. The five processes and a time quantum of 20 milliseconds, each process will get up ot 20 milliseconds every 100 milliseconds.

The performance of the RR algorithm depends heavily on the size of the time quantum. At one extreme, if the time quantum is extremely large, the RR policy is the same as the First Come First Serve policy. In contrast, if the time quantum is extremely small (say, 1 millisecond), the RR approach can result in a large number of context switches. Assume, for example. That we have only one process of 10 time units. If the quantum is 12 time units, the process finishes in less than 1 time quantum, with no overhead. If the quantum is 6 time units, however, the process requires 2 quanta, resulting in a context switch. If the time quantum is 1 time unit, then nine context switches will occur, slowing the execution of the process accordingly.Chart

Description automatically generated

Thus, we want the time quantum to be large with respect to the context switch time. If the context-switch time is approximately 10 percent of the time quantum, then about 10 percent of the CPU time will be spent in context switching. In practice, most modern systems have time quanta ranging from 10 to 100 milliseconds. The time required for a context switch is typically less than 10 microseconds; thus, the context-switch time is a small fraction of the time quantum.

1. **The processes are assumed to have arrived in the order P1, P2, P3, P4, P5, all at time 0.**

|  |  |  |
| --- | --- | --- |
| **Process** | **Burst Time** | **Priority** |
| P1 | 2 | 2 |
| P2 | 1 | 1 |
| P3 | 8 | 4 |
| P4 | 4 | 2 |
| P5 | 5 | 3 |
|  |  |  |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | |  | |  | |  | |  | |  | |  | |  | |  | |  | |  | |  | |  | |  | |  | |  | |  | |  | |  | |  | | |  | |  | |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | |  | |  | |  | |  | |  | |  | |  | |  | |  | |  | |  | |  |  | |  | |  |  |
| 0 | | 1 | | 2 | | 3 | | 4 | | 5 | | 6 | | 7 | | 8 | | 9 | | 10 | | 11 | | 12 | | 13 | | 14 | | 15 | | 16 | | 17 | | 18 | | 19 | | | 20 | |

Resource: <https://boonsuen.com/process-scheduling-solver>

* 1. **Draw four Gantt charts that illustrate the execution of these process using the following scheduling algorithms. Also calculate the turnaround time, and average waiting time:**

Turnaround Time:

* The interval from the time of submission of a process to the time of completion.
* Turnaround Time = Completion time – arrival time

Average Waiting Time:

* Time waiting in ready queue
  + 1. FCFS (First Come First Serve)

|  |  |  |
| --- | --- | --- |
| **Process** | **Burst Time** | **Priority** |
| P1 | 2 | 2 |
| P2 | 1 | 1 |
| P3 | 8 | 4 |
| P4 | 4 | 2 |
| P5 | 5 | 3 |

The Gantt chart for FCFS scheduling if the processes arrive at time 0 in the order: P1, P2, P3, P4, P5; is:

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | P1 | | | | P2 | | P3 | | | | | | | | | | | | | | | | P4 | | | | | | | | P5 | | | | | | | | | | |  | |  | | |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | |  | |  | |  | |  | |  | |  | |  | |  | |  | |  | |  | |  |  | |  | |  |  |
| 0 | | 1 | | 2 | | 3 | | 4 | | 5 | | 6 | | 7 | | 8 | | 9 | | 10 | | 11 | | 12 | | 13 | | 14 | | 15 | | 16 | | 17 | | 18 | | 19 | | | 20 | |

Turnaround Time:

Average Waiting Time:

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Process** | **Burst Time** | **Priority** | **Turnaround Time:** | | | | **Waiting Time** | | | |
| **FCFS** | **SJF** | **Non Pre-Emptive Priority** | **RR** | **FCFS** | **SJF** | **Non Pre-Emptive Priority** | **RR** |
| P1 | 2 | 2 | 2 |  |  |  | 0 |  |  |  |
| P2 | 1 | 1 | 3 |  |  |  | 2 |  |  |  |
| P3 | 8 | 4 | 11 |  |  |  | 3 |  |  |  |
| P4 | 4 | 2 | 15 |  |  |  | 11 |  |  |  |
| P5 | 5 | 3 | 20 |  |  |  | 15 |  |  |  |
| **Total** | 20 | - | 51 |  |  |  | 31 |  |  |  |
| **Linear Average** | 5 | - | 10.2 |  |  |  | 6.2 |  |  |  |

* + 1. SJF (Shortest Job First)

|  |  |  |
| --- | --- | --- |
| **Process** | **Burst Time** | **Priority** |
| P2 | 1 | 1 |
| P1 | 2 | 2 |
| P4 | 4 | 2 |
| P5 | 5 | 3 |
| P3 | 8 | 4 |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | P2 | | P1 | | | | P4 | | | | | | | | P5 | | | | | | | | | | P3 | | | | | | | | | | | | | | | | |  | |  | |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | |  | |  | |  | |  | |  | |  | |  | |  | |  | |  | |  | |  |  | |  | |  |  |
| 0 | | 1 | | 2 | | 3 | | 4 | | 5 | | 6 | | 7 | | 8 | | 9 | | 10 | | 11 | | 12 | | 13 | | 14 | | 15 | | 16 | | 17 | | 18 | | 19 | | | 20 | |

Turnaround Time:

Average Waiting Time:

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Process** | **Burst Time** | **Priority** | **Turnaround Time:** | | | | **Waiting Time** | | | |
| **FCFS** | **SJF** | **Non Pre-Emptive Priority** | **RR** | **FCFS** | **SJF** | **Non Pre-Emptive Priority** | **RR** |
| P1 | 2 | 2 | 2 | 3 |  |  | 0 | 1 |  |  |
| P2 | 1 | 1 | 3 | 1 |  |  | 2 | 0 |  |  |
| P3 | 8 | 4 | 11 | 20 |  |  | 3 | 12 |  |  |
| P4 | 4 | 2 | 15 | 7 |  |  | 11 | 3 |  |  |
| P5 | 5 | 3 | 20 | 12 |  |  | 15 | 7 |  |  |
| **Total** | 20 | - | 51 | 43 |  |  | 31 | 23 |  |  |
| **Linear Average** | 5 | - | 10.2 | 8.6 |  |  | 6.2 | 4.6 |  |  |

* + 1. Non pre-emptive priority (a larger priority number implies a higher priority)

|  |  |  |
| --- | --- | --- |
| **Process** | **Burst Time** | **Priority** |
| P3 | 8 | 4 |
| P5 | 5 | 3 |
| P1 | 2 | 2 |
| P4 | 4 | 2 |
| P2 | 1 | 1 |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | P3 | | | | | | | | | | | | | | | | P5 | | | | | | | | | | P1 | | | | P4 | | | | | | | | P2 | | |  | |  | |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | |  | |  | |  | |  | |  | |  | |  | |  | |  | |  | |  | |  |  | |  | |  |  |
| 0 | | 1 | | 2 | | 3 | | 4 | | 5 | | 6 | | 7 | | 8 | | 9 | | 10 | | 11 | | 12 | | 13 | | 14 | | 15 | | 16 | | 17 | | 18 | | 19 | | | 20 | |

Turnaround Time:

Average Waiting Time:

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Process** | **Burst Time** | **Priority** | **Turnaround Time:** | | | | **Waiting Time** | | | |
| **FCFS** | **SJF** | **Non Pre-Emptive Priority** | **RR** | **FCFS** | **SJF** | **Non Pre-Emptive Priority** | **RR** |
| P1 | 2 | 2 | 2 | 3 | 15 |  | 0 | 1 | 13 |  |
| P2 | 1 | 1 | 3 | 1 | 20 |  | 2 | 0 | 19 |  |
| P3 | 8 | 4 | 11 | 20 | 8 |  | 3 | 12 | 0 |  |
| P4 | 4 | 2 | 15 | 7 | 19 |  | 11 | 3 | 15 |  |
| P5 | 5 | 3 | 20 | 12 | 13 |  | 15 | 7 | 8 |  |
| **Total** | 20 | - | 51 | 43 | 75 |  | 31 | 23 | 55 |  |
| **Linear Average** | 5 | - | 10.2 | 8.6 | 15 |  | 6.2 | 4.6 | 11 |  |

* + 1. RR (Round Robin) (quantum = 2)

Round 1:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Process** | **Burst Time** | **Quantum** | **Remaining Time** | **Priority** |
| P1 | 2 | 2 |  | 2 |
| P2 | 1 | 2 | (no time left – can’t be negative) | 1 |
| P3 | 8 | 2 |  | 4 |
| P4 | 4 | 2 |  | 2 |
| P5 | 5 | 2 |  | 3 |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | P1 | | | | P2 | | P3 | | | | P4 | | | | P5 | | | |  | |  | |  | |  | |  | |  | |  | |  | |  | |  | |  | | |  | |  | |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | |  | |  | |  | |  | |  | |  | |  | |  | |  | |  | |  | |  |  | |  | |  |  |
| 0 | | 1 | | 2 | | 3 | | 4 | | 5 | | 6 | | 7 | | 8 | | 9 | | 10 | | 11 | | 12 | | 13 | | 14 | | 15 | | 16 | | 17 | | 18 | | 19 | | | 20 | |

Round 2:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Process** | **Burst Time** | **Quantum** | **Remaining Time** | **Priority** |
| P1 | 0 | 2 | 0 | 2 |
| P2 | 0 | 2 | 0 | 1 |
| P3 | 6 | 2 |  | 4 |
| P4 | 2 | 2 |  | 2 |
| P5 | 3 | 2 |  | 3 |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | P1 | | | | P2 | | P3 | | | | P4 | | | | P5 | | | | P3 | | | | P4 | | | | P5 | | | |  | |  | |  | |  | |  | | |  | |  | |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | |  | |  | |  | |  | |  | |  | |  | |  | |  | |  | |  | |  |  | |  | |  |  |
| 0 | | 1 | | 2 | | 3 | | 4 | | 5 | | 6 | | 7 | | 8 | | 9 | | 10 | | 11 | | 12 | | 13 | | 14 | | 15 | | 16 | | 17 | | 18 | | 19 | | | 20 | |

Round 3:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Process** | **Burst Time** | **Quantum** | **Remaining Time** | **Priority** |
| P1 | 0 | 2 | 0 | 2 |
| P2 | 0 | 2 | 0 | 1 |
| P3 | 4 | 2 |  | 4 |
| P4 | 0 | 2 |  | 2 |
| P5 | 1 | 2 |  | 3 |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | P1 | | | | P2 | | P3 | | | | P4 | | | | P5 | | | | P3 | | | | P4 | | | | P5 | | | | P3 | | | | P5 | |  | |  | | |  | |  | |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | |  | |  | |  | |  | |  | |  | |  | |  | |  | |  | |  | |  |  | |  | |  |  |
| 0 | | 1 | | 2 | | 3 | | 4 | | 5 | | 6 | | 7 | | 8 | | 9 | | 10 | | 11 | | 12 | | 13 | | 14 | | 15 | | 16 | | 17 | | 18 | | 19 | | | 20 | |

Round 4:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Process** | **Burst Time** | **Quantum** | **Remaining Time** | **Priority** |
| P1 | 0 | 2 | 0 | 2 |
| P2 | 0 | 2 | 0 | 1 |
| P3 | 2 | 2 |  | 4 |
| P4 | 0 | 2 |  | 2 |
| P5 | 1 | 2 |  | 3 |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | P1 | | | | P2 | | P3 | | | | P4 | | | | P5 | | | | P3 | | | | **P4** | | | | P5 | | | | P3 | | | | **P5** | | P3 | | | | |  | |  | |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | |  | |  | |  | |  | |  | |  | |  | |  | |  | |  | |  | |  |  | |  | |  |  |
| 0 | | 1 | | 2 | | **3** | | 4 | | 5 | | 6 | | 7 | | 8 | | 9 | | 10 | | 11 | | 12 | | **13** | | 14 | | 15 | | 16 | | 17 | | **18** | | 19 | | | 20 | |

Turnaround Time:

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Process** | **Burst Time** | **Priority** | **Turnaround Time:** | | | | **Waiting Time** | | | |
| **FCFS** | **SJF** | **Non Pre-Emptive Priority** | **RR** | **FCFS** | **SJF** | **Non Pre-Emptive Priority** | **RR** |
| P1 | 2 | 2 | 2 | 2 | 15 | 2 | 0 | 1 | 13 |  |
| P2 | 1 | 1 | 3 | 1 | 20 | 3 | 2 | 0 | 19 |  |
| P3 | 8 | 4 | 11 | 20 | 8 | 20 | 3 | 12 | 0 |  |
| P4 | 4 | 2 | 15 | 7 | 19 | 13 | 11 | 3 | 15 |  |
| P5 | 5 | 3 | 20 | 12 | 13 | 18 | 15 | 7 | 8 |  |
| **Total** | 20 | - | 51 | 42 | 75 | 56 | 31 | 23 | 55 |  |
| **Linear Average** | 5 | - | 10.2 | 8.4 | 15 | 11.2 | 6.2 | 4.6 | 11 |  |

Average Waiting Time:

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | P1 | | | | P2 | | P3 | | | | P4 | | | | P5 | | | | P3 | | | | P4 | | | | P5 | | | | P3 | | | | P5 | | P3 | | | | |  | |  | |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | |  | |  | |  | |  | |  | |  | |  | |  | |  | |  | |  | |  |  | |  | |  |  |
| 0 | | 1 | | 2 | | 3 | | 4 | | 5 | | 6 | | 7 | | 8 | | 9 | | 10 | | 11 | | 12 | | 13 | | 14 | | 15 | | 16 | | 17 | | 18 | | 19 | | | 20 | |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | P1 | | | | P2 | | P3 | | | | P4 | | | | P5 | | | | P3 | | | | P4 | | | | P5 | | | | P3 | | P5 | | | | P3 | | | | |  | |  | |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | |  | |  | |  | |  | |  | |  | |  | |  | |  | |  | |  | |  |  | |  | |  |  |
| 0 | | 1 | | 2 | | 3 | | 4 | | 5 | | 6 | | 7 | | 8 | | 9 | | 10 | | 11 | | 12 | | 13 | | 14 | | 15 | | 16 | | 17 | | 18 | | 19 | | | 20 | |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | P1 | | | | P2 | | P3 | | | | P4 | | | | P5 | | | | P3 | | | | P4 | | | | P5 | | | | P3 | | | | P5 | | P3 | | | | |  | |  | |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | |  | |  | |  | |  | |  | |  | |  | |  | |  | |  | |  | |  |  | |  | |  |  |
| 0 | | 1 | | 2 | | 3 | | 4 | | 5 | | 6 | | 7 | | 8 | | 9 | | 10 | | 11 | | 12 | | 13 | | 14 | | 15 | | 16 | | 17 | | 18 | | 19 | | | 20 | |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | P1 | | | | P2 | | P3 | | | | P4 | | | | P5 | | | | P3 | | | | P4 | | | | P5 | | | | P3 | | | | P5 | | P3 | | | | |  | |  | |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | |  | |  | |  | |  | |  | |  | |  | |  | |  | |  | |  | |  |  | |  | |  |  |
| 0 | | 1 | | 2 | | 3 | | 4 | | 5 | | 6 | | 7 | | 8 | | 9 | | 10 | | 11 | | 12 | | 13 | | 14 | | 15 | | 16 | | 17 | | 18 | | 19 | | | 20 | |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | P1 | | | | P2 | | P3 | | | | P4 | | | | P5 | | | | P3 | | | | P4 | | | | P5 | | | | P3 | | | | P5 | | P3 | | | | |  | |  | |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | |  | |  | |  | |  | |  | |  | |  | |  | |  | |  | |  | |  |  | |  | |  |  |
| 0 | | 1 | | 2 | | 3 | | 4 | | 5 | | 6 | | 7 | | 8 | | 9 | | 10 | | 11 | | 12 | | 13 | | 14 | | 15 | | 16 | | 17 | | 18 | | 19 | | | 20 | |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | P1 | | | | P2 | | P3 | | | | P4 | | | | P5 | | | | P3 | | | | P4 | | | | P5 | | | | P3 | | | | P5 | | P3 | | | | |  | |  | |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | |  | |  | |  | |  | |  | |  | |  | |  | |  | |  | |  | |  |  | |  | |  |  |
| 0 | | 1 | | 2 | | 3 | | 4 | | 5 | | 6 | | 7 | | 8 | | 9 | | 10 | | 11 | | 12 | | 13 | | 14 | | 15 | | 16 | | 17 | | 18 | | 19 | | | 20 | |

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Process** | **Burst Time** | **Priority** | **Turnaround Time:** | | | | **Waiting Time** | | | |
| **FCFS** | **SJF** | **Non Pre-Emptive Priority** | **RR** | **FCFS** | **SJF** | **Non Pre-Emptive Priority** | **RR** |
| P1 | 2 | 2 | 2 | 3 | 15 | 2 | 0 | 1 | 13 | 0 |
| P2 | 1 | 1 | 3 | 1 | 20 | 3 | 2 | 0 | 19 | 2 |
| P3 | 8 | 4 | 11 | 20 | 8 | 20 | 3 | 12 | 0 | 12 |
| P4 | 4 | 2 | 15 | 7 | 19 | 13 | 11 | 3 | 15 | 9 |
| P5 | 5 | 3 | 20 | 12 | 13 | 18 | 15 | 7 | 8 | 13 |
| **Total** | 20 | - | 51 | 43 | 75 | 56 | 31 | 23 | 55 | 36 |
| **Linear Average** | 5 | - | 10.2 | 8.6 | 15 | 11.2 | 6.2 | 4.6 | 11 | 7.2 |

Notes – Typically:

* RR has a higher turnaround time compared to SJF: TRUE HERE
* RR has lower (better) average waiting time than SJF: **NOT** TRUE HERE
  + **Reason**: Due to the small quantum, there is more context switching which causes more waiting time for the RR scheduling. Increasing the quantum will result in a shorter waiting time for RR scheduling.
  1. **Which of the algorithms results in the minimum average waiting time (over all processes)?**
     1. FCFS (First Come First Serve)
     2. SJF (Shortest Job First)
     3. Non pre-emptive priority (a larger priority number implies a higher priority)
     4. RR (Round Robin) (quantum = 2)

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Process** | **Burst Time** | **Priority** | **Turnaround Time:** | | | | **Waiting Time** | | | |
| **FCFS** | **SJF** | **Non Pre-Emptive Priority** | **RR** | **FCFS** | **SJF** | **Non Pre-Emptive Priority** | **RR** |
| P1 | 2 | 2 | 2 | 3 | 15 | 2 | 0 | 1 | 13 | 0 |
| P2 | 1 | 1 | 3 | 1 | 20 | 3 | 2 | 0 | 19 | 2 |
| P3 | 8 | 4 | 11 | 20 | 8 | 20 | 3 | 12 | 0 | 13 |
| P4 | 4 | 2 | 15 | 7 | 19 | 13 | 11 | 3 | 15 | 9 |
| P5 | 5 | 3 | 20 | 12 | 13 | 18 | 15 | 7 | 8 | 12 |
| **Total** | 20 | - | 51 | 43 | 75 | 56 | 31 | 23 | 55 | 36 |
| **Linear Average** | - | - | 10.2 | 8.6 | 15 | 11.2 | 6.2 | 4.6 | 11 | 7.2 |

**See notes in previous question as to why RR does NOT have a better waiting time than SJF.**

# Chapter Review

## Chapter 1

* An operating system is software that manages the computer hardware, as well as providing an environment for application programs to run.
* Interrupts are a key way in which hardware interacts with the operating system. A hardware device triggers an interrupt by sending a signal to the CPU to alert the CPU that some event requires attention. The interrupt is managed by the interrupt handler.
* For a computer to do its job of executing programs, the programs must be in main memory, which is the only large storage area that the processor can access directly.
* The main memory is usually a volatile storage device that loses its contents when power is turned off or lost.
* Non-volatile storage is an extension of main memory and is capable of holding large quantities of data permanently.
* The most common non-volatile storage device is a hard disk, which can provide storage of both programs and data.
* The wide variety of storage systems in a computer system can be organized in a hierarchy according to speed and cost. The higher levels are expensive, but they are fast. As we move down the hierarchy, the cost per bit generally decreases, whereas the access time generally increases.
* Modern computer architectures are multiprocessor systems in which each CPU contains several computing cores.
* To best utilize the CPU, modern operating systems employ multiprogramming, which allows several jobs to be in memory at the same time, thus ensuring that the CPU always has a job to execute.
* Multitasking is an extension of multiprogramming wherein CPU scheduling algorithms rapidly switch between processes, providing users with a fast response time.
* To prevent user programs from interfering with the proper operation of the system, the system hardware has two modes: user mode and kernel mode.
* Various instructions are privileged and can be executed only in kernel mode. Examples include the instructions to switch to kernel mode, I/O control, timer management, and interrupt management.

## Chapter 2

## Chapter 3

## Chapter 4