COMP 3511
Operating Systems

Lab #4 Review

Q. 1

- The _____ of a process contains temporary data such as function parameters, return addresses, and local variables.
- A) text section
- B) data section
- C) program counter
- D) stack

Q. 1

- The _____ of a process contains temporary data such as function parameters, return addresses, and local variables.
- A) text section
- B) data section
- C) program counter
- D) stack

- D

Q. 2

- A process control block ____.
- A) includes information on the process's state
- B) stores the address of the next instruction to be processed by a different process
- C) determines which process is to be executed next
- D) is an example of a process queue

Q. 2

- A process control block ____.
- A) includes information on the process's state
- B) stores the address of the next instruction to be processed by a different process
- C) determines which process is to be executed next
- D) is an example of a process queue

- A

Q. 3

- Imagine that a host with IP address 150.55.66.77 wishes to download a file from the web server at IP address 202.28.15.123. Select a valid socket pair for a connection between this pair of hosts.
- A) 150.55.66.77:80 and 202.28.15.123:80
- B) 150.55.66.77:150 and 202.28.15.123:80
- C) 150.55.66.77:2000 and 202.28.15.123:80
- D) 150.55.66.77:80 and 202.28.15.123:3500

Q. 3

- Imagine that a host with IP address 150.55.66.77 wishes to download a file from the web server at IP address 202.28.15.123. Select a valid socket pair for a connection between this pair of hosts.
- A) 150.55.66.77:80 and 202.28.15.123:80
- B) 150.55.66.77:150 and 202.28.15.123:80
- C) 150.55.66.77:2000 and 202.28.15.123:80
- D) 150.55.66.77:80 and 202.28.15.123:3500

- C

Q. 4

- Child processes inherit UNIX ordinary pipes from their parent process because:
- A) The pipe is part of the code and children inherit code from their parents.
- B) A pipe is treated as a file descriptor and child processes inherit open file descriptors from their parents.
- C) The STARTUPINFO structure establishes this sharing.
- D) All IPC facilities are shared between the parent and child processes.

Q. 4

- Child processes inherit UNIX ordinary pipes from their parent process because:
- A) The pipe is part of the code and children inherit code from their parents.
- B) A pipe is treated as a file descriptor and child processes inherit open file descriptors from their parents.
- C) The STARTUPINFO structure establishes this sharing.
- D) All IPC facilities are shared between the parent and child processes.
- B

Q. 5

- The _________ application is the application appearing on the display screen of a mobile device.
- A) main
- B) background
- C) display
- D) foreground

Q. 5

- The _________ application is the application appearing on the display screen of a mobile device.
- A) main
- B) background
- C) display
- D) foreground
- D

Q. 6

- A process that has terminated, but whose parent has not yet called wait(), is known as a _________ process.
- A) zombie
- B) orphan
- C) terminated
- D) init

Q. 6

- A process that has terminated, but whose parent has not yet called wait(), is known as a ________ process.
- A) zombie
- B) orphan
- C) terminated
- D) init

- A

Q. 7

- The ________ process is assigned as the parent to orphan processes.
- A) zombie
- B) init
- C) main
- D) renderer

Q. 7

- The ________ process is assigned as the parent to orphan processes.
- A) zombie
- B) init
- C) main
- D) renderer
- B

Q. 8

- Cancellation points are associated with _____ cancellation.
- A) asynchronous
- B) deferred
- C) synchronous
- D) non-deferred

Q. 8

- Cancellation points are associated with _____ cancellation.
- A) asynchronous
- B) deferred
- C) synchronous
- D) non-deferred

- B

Q. 9

- Which of the following would be an acceptable signal handling scheme for a multithreaded program?
- A) Deliver the signal to the thread to which the signal applies.
- B) Deliver the signal to every thread in the process.
- C) Deliver the signal to only certain threads in the process.
- D) All of the above

Q. 9

- Which of the following would be an acceptable signal handling scheme for a multithreaded program?
- A) Deliver the signal to the thread to which the signal applies.
- B) Deliver the signal to every thread in the process.
- C) Deliver the signal to only certain threads in the process.
- D) All of the above

- D

Q. 10

- A _____ provides an API for creating and managing threads.
- A) set of system calls
- B) multicore system
- C) thread library
- D) multithreading model

Q. 10

- A ____ provides an API for creating and managing threads.
- A) set of system calls
- B) multicore system
- C) thread library
- D) multithreading model
- C

Q. 11

- Ordinarily the exec() system call follows the fork(). Explain what would happen if a programmer were to inadvertently place the call to exec() before the call to fork().

  - Because exec() overwrites the process, we would never reach the call to fork() and hence, no new processes would be created. Rather, the program specified in the parameter to exec() would be run instead.

Q. 12

- How can deferred cancellation ensure that thread termination occurs in an orderly manner as compared to asynchronous cancellation?
  - In asynchronous cancellation, the thread is immediately cancelled in response to a cancellation request. There is no insurance that it did not quit in the middle of a data update or other potentially dangerous situation. In deferred cancellation, the thread checks whether or not it should terminate. This way, the thread can be made to cancel at a convenient time.

Q. 13

- Distinguish between data and task parallelism.

  - Data parallelism involves distributing subsets of the same data across multiple computing cores and performing the same operation on each core. Task parallelism involves distributing tasks across the different computing cores where each task is performing a unique operation.

Q 14

- _____ is the number of processes that are completed per time unit.
- A) CPU utilization
- B) Response time
- C) Turnaround time
- D) Throughput

Q 14

- _____ is the number of processes that are completed per time unit.
- A) CPU utilization
- B) Response time
- C) Turnaround time
- D) Throughput
- D

Q 15

- _____ scheduling is approximated by predicting the next CPU burst with an exponential average of the measured lengths of previous CPU bursts.
- A) Multilevel queue
- B) RR
- C) FCFS
- D) SJF

Q 15

- _____ scheduling is approximated by predicting the next CPU burst with an exponential average of the measured lengths of previous CPU bursts.
- A) Multilevel queue
- B) RR
- C) FCFS
- D) SJF

- D

Q 16

- Which of the following scheduling algorithms must be nonpreemptive?
- A) SJF
- B) RR
- C) FCFS
- D) priority algorithms

Q 16

- Which of the following scheduling algorithms must be nonpreemptive?
- A) SJF
- B) RR
- C) FCFS
- D) priority algorithms

- C

Q. 17

- What is the main difference between preemptive and non-preemptive scheduling.
  - **Preemptive scheduling** allows a process to be interrupted in the midst of its execution, taking the CPU away and allocating it to another process.
  - **Non-preemptive scheduling** ensures that a process relinquishes control of the CPU only when it finishes with its current CPU burst.

Q. 18

- Why is it important for the scheduler to distinguish I/O-bound programs from CPU-bound programs?

- **I/O-bound programs**
  - Little computation
  - Many I/O operations
  - Do not use up their CPU quantum
- **CPU-bound programs**
  - Much computation
  - Few I/O operations
  - Use up their CPU quantum

To use resources better

Give higher priority to I/O-bound programs

Q. 19

- Consider a system running
  - 1 CPU-bound task
  - 10 I/O-bound tasks
    - each issue an I/O operation once for every ms of CPU computing
    - each I/O operation takes 10 ms to complete
  - context switching overhead = 0.1ms
  - all processes are long-running tasks

- What is the CPU utilization for a round-robin scheduler when:
  - (a) The time quantum is 1 ms

Q. 19

- Consider a system running
  - 1 CPU-bound task
  - 10 I/O-bound tasks
    - each issue an I/O operation once for every ms of CPU computing
    - each I/O operation takes 10 ms to complete
  - context switching overhead = 0.1ms
  - all processes are long-running tasks

- What is the CPU utilization for a round-robin scheduler when:
  - (a) The time quantum is 1 ms

Time quantum 1 ms	Switching 0.1 ms

CPU utilization:

$1/1.1 = 91\%$

Q. 19

- Consider a system running
  - 1 CPU-bound task
  - 10 I/O-bound tasks
    - each issue an I/O operation once for every ms of CPU computing
    - each I/O operation takes 10 ms to complete
  - context switching overhead = 0.1ms
  - all processes are long-running tasks

- What is the CPU utilization for a round-robin scheduler when:
  - (b) The time quantum is 10 ms

Q. 19

- Consider a system running
  - 1 CPU-bound task
  - 10 I/O-bound tasks
    - each issue an I/O operation once for every ms of CPU computing
    - each I/O operation takes 10 ms to complete
  - context switching overhead = 0.1ms
  - all processes are long-running tasks

- What is the CPU utilization for a round-robin scheduler when:
  - (b) The time quantum is 10 ms

I/O-bound tasks	Use 1 ms	Switching 0.1 ms	Time to cycle 10 I/O-bound tasks = 1.1 x 10
CPU-bound tasks	Time quantum 10 ms	Switching 0.1 ms	Time to cycle 1 CPU-bound tasks = 10.1

CPU utilization:

20/21.1= 94%

Q. 20

- How the following pairs of scheduling criteria conflict in certain settings
  - CPU utilization and response time
    - CPU utilization is increased if the overheads associated with context switching is minimized. The context switching overheads could be lowered by performing context switches infrequently. This could, however, result in increasing the response time for processes
  - Average turnaround time and maximum waiting time
    - Average turnaround time is minimized by executing the shortest tasks first. Such a scheduling policy could, however, starve long-running tasks and thereby increase their waiting time

Q. 20

- How the following pairs of scheduling criteria conflict in certain settings
  - I/O device utilization and CPU utilization
    - CPU utilization is maximized by running long-running CPU-bound tasks without performing context switches. I/O device utilization is maximized by scheduling I/O-bound jobs as soon as they become ready to run, thereby incurring the overheads of context switches

Q. 21

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

Q. 21

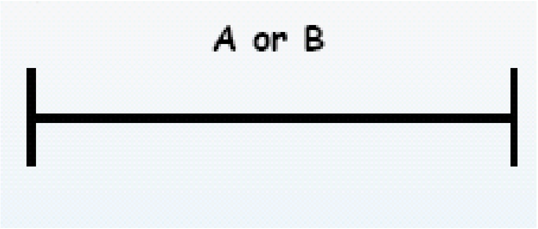
- Which of the following scheduling algorithms could result in starvation
  - a. First-come, first-served
  - b. Shortest job first
  - c. Round robin
  - d. Priority

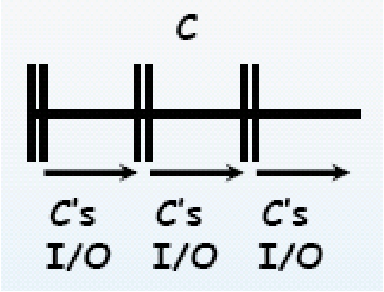
- Answer: Shortest job first and priority-based scheduling algorithms could result in starvation

Q. 22

- **Example to illustrate benefits of SRTF** (*Shortest Remaining Time First*)

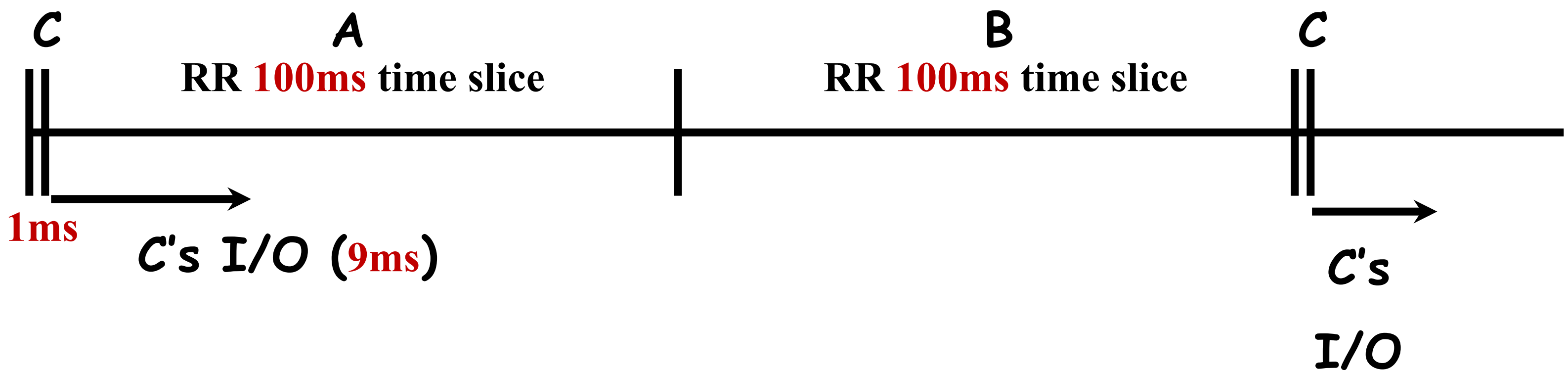
- Three jobs:
  - A, B: both CPU bound, each run for one week
  - C: I/O bound, loop 1ms CPU, 9ms disk I/O, and continue
  - If only one at a time, C uses 90% of the disk, A or B could use 100% of the CPU





Q. 23

- Compute the Disk utilization:
  - RR with **100ms time slice**

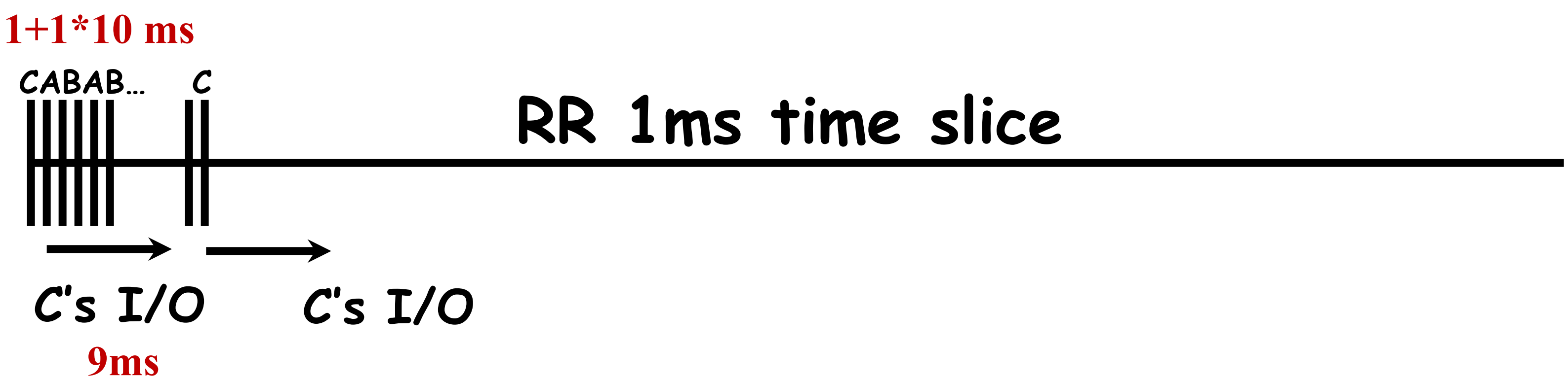


Disk Utilization:

9/(1+100+100) = 9/201 ~ 4.5%

Q. 23

- Compute the Disk utilization:
  - RR with **1ms time slice**

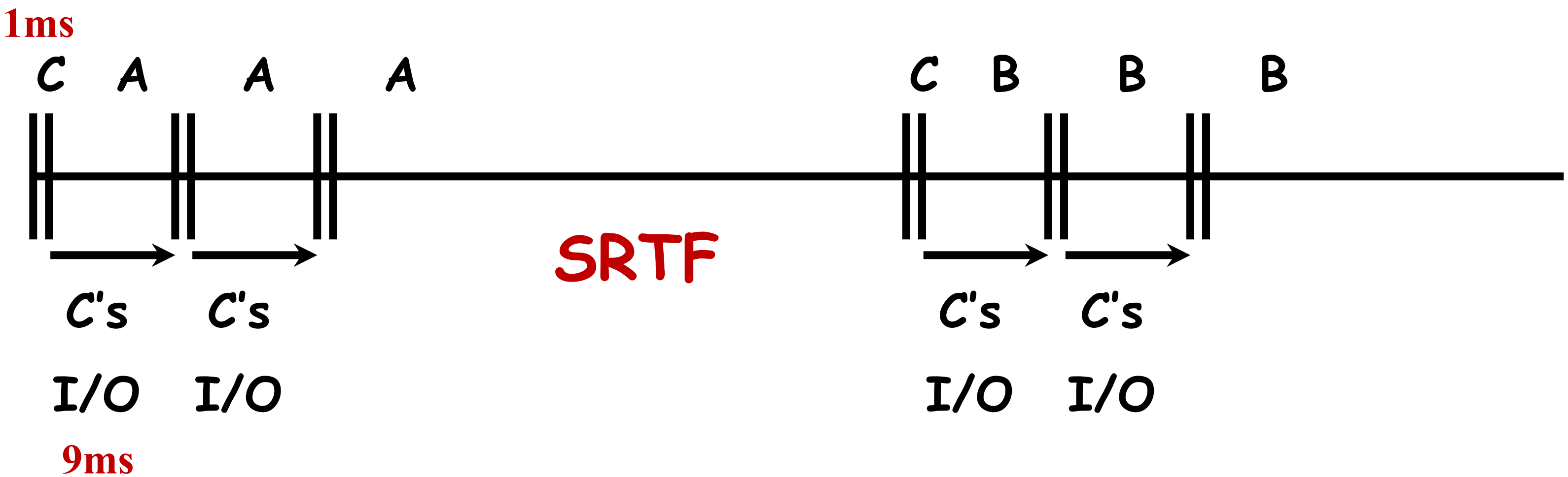


Disk Utilization:

9/(1+1*10) = 82%, ~90% but lots of wakeups!

Q. 23

- Compute the Disk utilization:
  - **SRTF**



Disk Utilization:

9/(1+9) = 90%

Q. 24

- Suppose that 4 processes arrive for execution at the times indicated by their "*Arrival Time*", respectively.
- Each process will run the amount of time indicated by its "*Burst Time*".

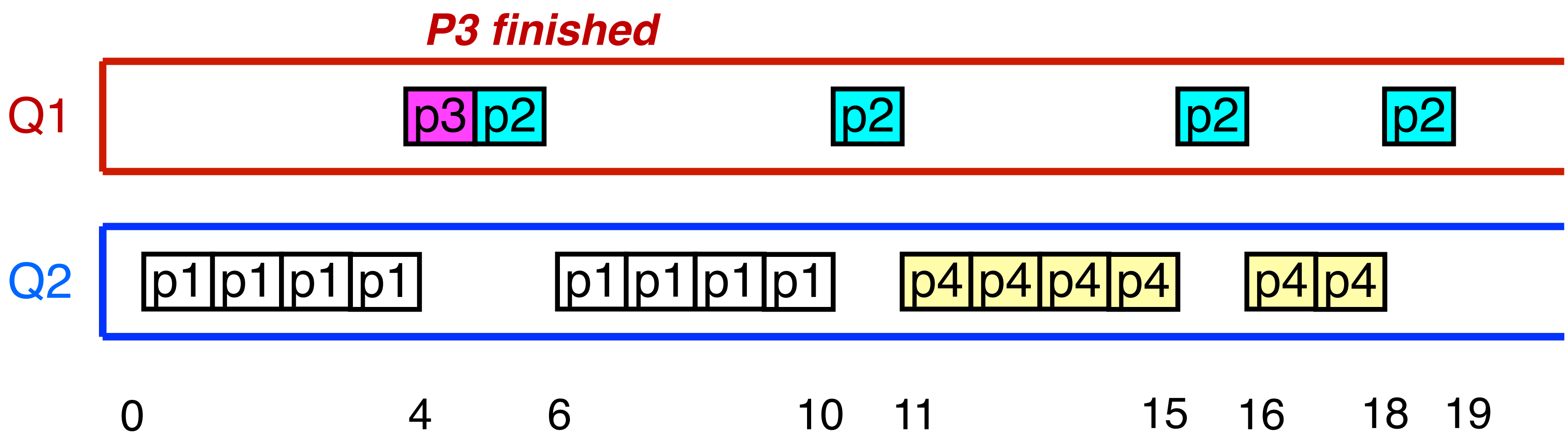
Process	Arrival Time	Burst Time	Type
P1	0.0	8	batch
P2	0.4	4	interactive
P3	1.0	1	interactive
P4	7.6	6	batch

Q. 24

- A multilevel queue (but not a multilevel feedback queue) is used for scheduling:
  - The *1st* queue is for interactive processes only (e.g., P2 and P3), and uses RR with q = 1 (time unit).
  - The *2nd* queue is for batch processes (e.g., P1 and P4), and uses FCFS with up to 4 (time units) each time when such processes are in the CPU.
  - Scheduling between queues
    - 1 process is dispatched from the batch queue, followed by 2 processes dispatched from the interactive queue, etc.

Q. 24

- Calculate the average turnaround time:
  - Scheduling sequence as follows:



~~**average turnaround time:**~~

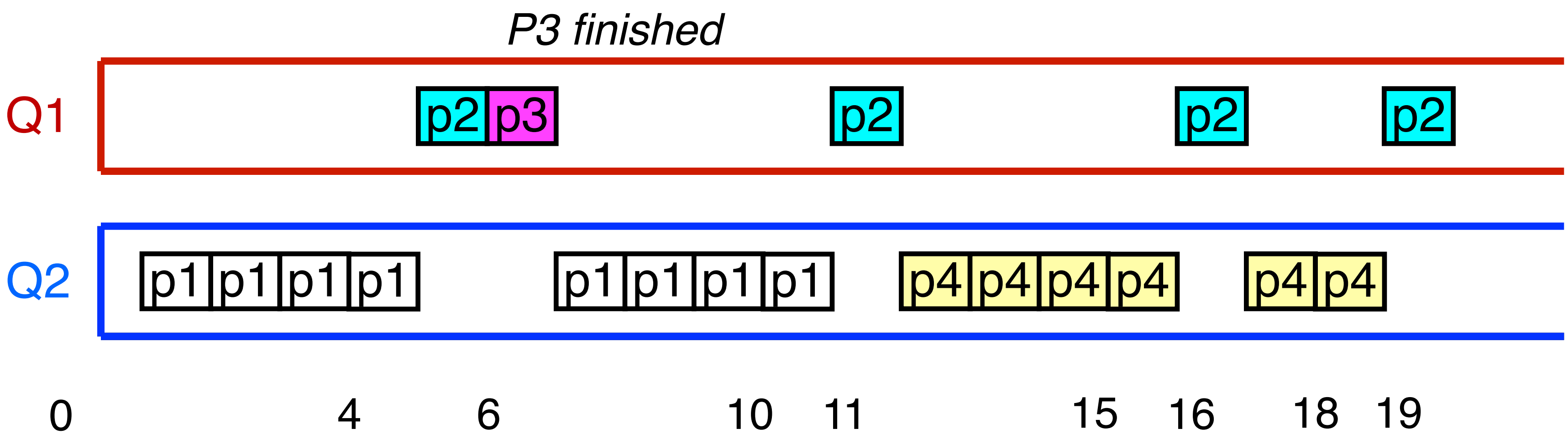
~~**(10 + 19 + 6 + 18) / 4 = 13.25**~~

Correct? NO

Q. 24

- Calculate the average turnaround time:

Process	Arrival Time	Burst Time	Type
P1	**0.0**	**8**	batch
P2	**0.4**	**4**	interactive
P3	**1.0**	**1**	interactive
P4	**7.6**	**6**	batch



Note that turnaround time = waiting time + burst

So average turnaround time:

[10 + (19-0.4) + (6-1) + (18-7.6)] / 4 = 11

Q. 25

- What role does the dispatcher play in CPU scheduling?
- Ans: The dispatcher gives control of the CPU to the process selected by the short-term scheduler. To perform this task, a context switch, a switch to user mode, and a jump to the proper location in the user program are all required. The dispatch should be made as fast as possible. The time lost to the dispatcher is termed dispatch latency.

Q. 26

- Explain the difference between response time and turnaround time. These times are both used to measure the effectiveness of scheduling schemes.
- Ans: Turnaround time is the sum of the periods that a process is spent waiting to get into memory, waiting in the ready queue, executing on the CPU, and doing I/O. Turnaround time essentially measures the amount of time it takes to execute a process. Response time, on the other hand, is a measure of the time that elapses between a request and the first response produced.

Q. 27

- What effect does the size of the time quantum have on the performance of an RR algorithm?
- Ans: At one extreme, if the time quantum is extremely large, the RR policy is the same as the FCFS policy. If the time quantum is extremely small, the RR approach is called processor sharing and creates the appearance that each of n processes has its own processor running at 1/n the speed of the real processor.

Q. 28

- Explain the process of starvation and how aging can be used to prevent it.
- Ans: Starvation occurs when a process is ready to run but is stuck waiting indefinitely for the CPU. This can be caused, for example, when higher-priority processes prevent low-priority processes from ever getting the CPU. Aging involves gradually increasing the priority of a process so that a process will eventually achieve a high enough priority to execute if it waited for a long enough period of time.

Q. 29

- Describe two general approaches to load balancing.
- Ans: With push migration, a specific task periodically checks the load on each processor and — if it finds an imbalance—evenly distributes the load by moving processes from overloaded to idle or less-busy processors. Pull migration occurs when an idle processor pulls a waiting task from a busy processor. Push and pull migration are often implemented in parallel on load-balancing systems.