Unit – II
Process Management

Objectives:

- ➢ Students will be able to develop the concepts of process management techniques

Syllabus:

Process, process state, process controls block (PCB),

Process scheduling- scheduling queues, schedulers, context switch, scheduling criteria, scheduling algorithms, Operations on processes, Inter process communication.

Outcomes:

Students will be able to

- Learn the Process, Process state diagram and various fields in process control block.(PCB)
- Explain the terms scheduling queues, schedulers, context switch and scheduling criteria.
- Differentiate Preemptive and non preemptive scheduling algorithms.
- Explain various techniques used for inter process communication.
- Know the benefits of multi-threading and models.

Learning Material

2.1. Process:

- A program in execution is called a process.
- A process will need certain resources such as CPU time, memory, files and I/O devices to accomplish its task.
- These resources are allocated to the process either when it is created or while it is executing.
- An operating system executes a variety of programs: Batch system jobs, Time shared systems, user programs or tasks
- Generally the terms ***job*** and ***process*** are almost the same.
- Process execution must progress in sequential fashion

A process includes:

- program counter
- stack
- data section

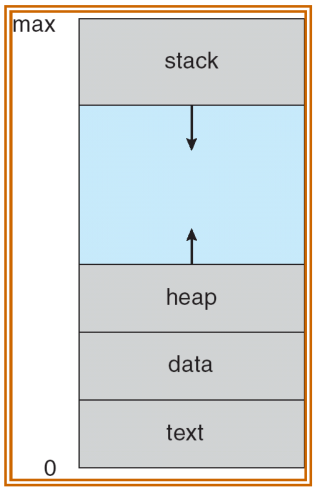


Fig: Process in memory

- A process is a program under execution.
- It generally includes process stack, containing temporary data (such as sub routine parameters, return addresses, temporary variables) and data section containing global variables.

- A program is a passive entity such as the contents of files stored on a disk.
- A Process is an active entity with a program counter, specifying the next instruction to execute and set of associated resources.

2.2. Process State:

- As a process executes, it changes state.
- The state of process is defined in part by the current activity of that process.
- Each process may be in one of the following states.

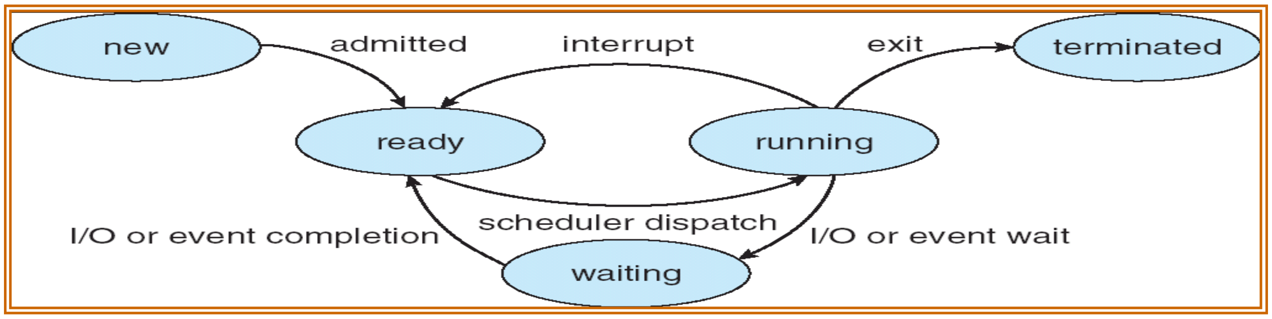


Fig: Process State Diagram

New: The process is being created.

Running: Instructions are being executed.

Waiting: The process is waiting for some event to occur (such as I/O completion or reception of a signal).

Ready: The process is waiting to be assigned to the processor.

Terminated: The process has finished execution.

- ➢ At any point of time only one process can be running. Many processes may be ready and waiting.

2.3. Process Control Block:

- ➢ Each process is represented in the operating system by a process control block also called task control block.

Information associated with each process

- Process state
- Program counter
- CPU registers
- CPU scheduling information
- Memory-management information
- Accounting information
- I/O status information

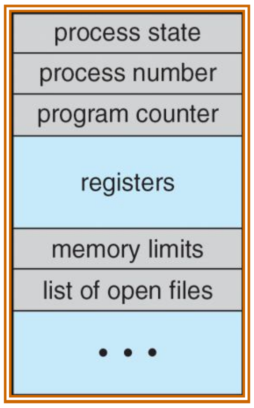


Fig: Process Control Block (PCB)

It contains many pieces of information associated with a specific process, including these:

- **Process State:** The state may be new, ready, running, and waiting, halted and so on…
- **Program Counter:** The counter indicates the address of next instruction to be executed for this process.
- **CPU Registers:** The registers vary in number and type, depending on computer architecture. They include accumulators, index registers, stack pointers and general purpose register plus any condition code information.

- **CPU Scheduling Information:** The information includes a process priority, pointers to scheduling queues and any other scheduling parameters
- **Memory Management Information :** This include information such as the values of the base and limit registers, the page tables or segment tables depending on the memory system used by operating system
- **Accounting Information:** Thi**s includes** the amount of CPU and real time used, time limits, account numbers, job or process numbers and so on…
- **I/O Status Information:** This includes the list of I/O devices such as tape drivers allocated to this process, list of open files and so on….

The simply serve as the repository for any information that may vary from process to process.

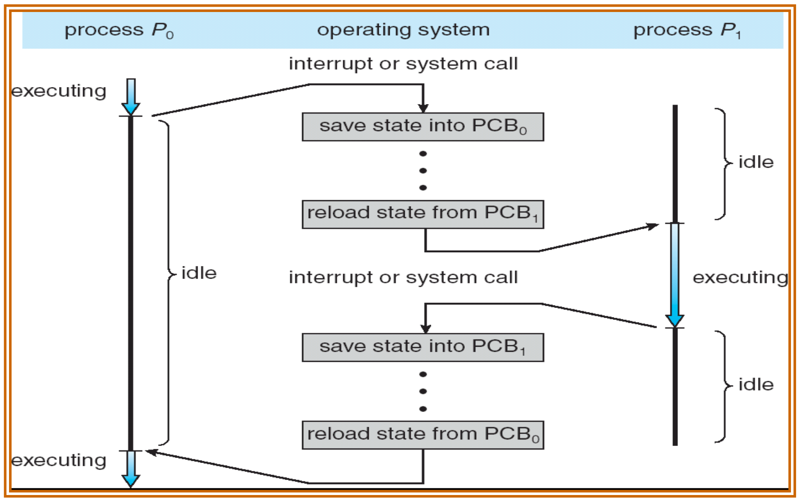


Fig: CPU switches from process to process

2.4. <u>Process Scheduling:</u>

- The objective of multiprogramming is to have some process running at all times.
- To maximize CPU utilization for any processor system, there will never be more than one running process.
- If there are more process the rest will have to wait until the CPU is free and can be rescheduled.

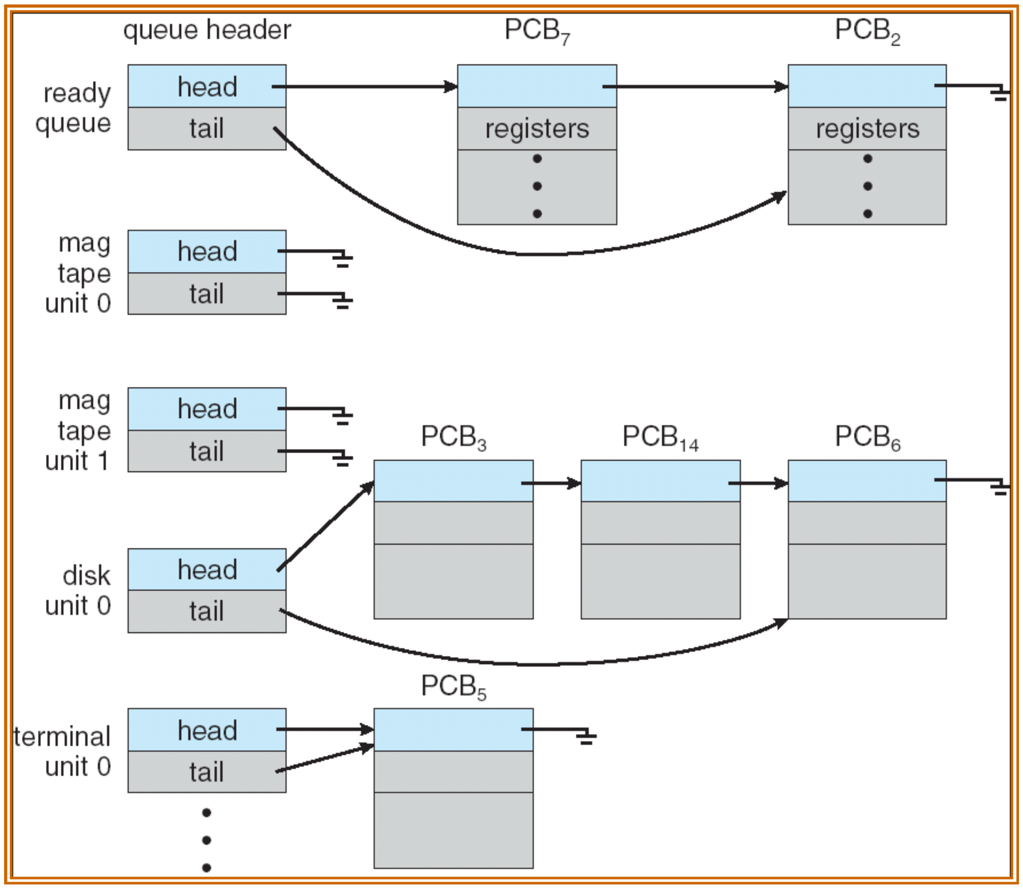


Fig: The ready queue and various I/O device queues

2.4.1 Scheduling Queues:

- **Job Queue:** As processes enter the system they are put into a job queue this queue consists of all process in the system
- **Ready queue:** The processes that are residing in main memory and are ready and waiting to execute are kept a list called the ready queue. This queue is generally stored as a linked list. A ready queue header will contain pointers to the first and last PCB's in the list. Each PCB has a pointer field that points to the next process in the ready queue.
- **Device Queue:** A list of process waiting for a particular I/O device is called a device queue.

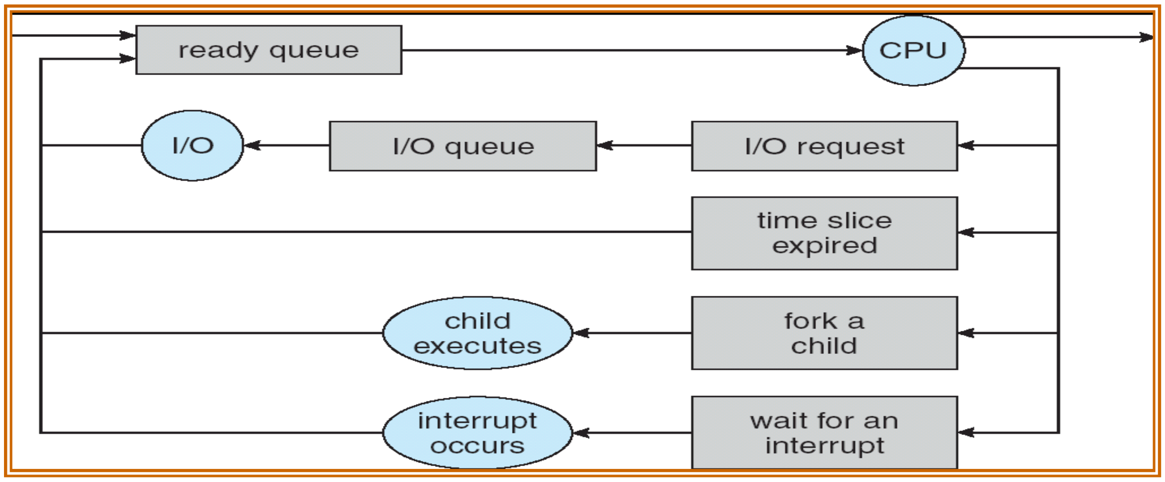


Fig: Queuing diagram representation of process scheduling.

- In queuing diagram each rectangular box represents a queue.
- The circles represent the resources that serve the queues
- The arrows indicate the flow of processes in the system.

2.4.2 Schedulers:

➢ **Long-term Scheduler:**

- It is also known as job scheduler.
- It selects processes from this pool and loads them into memory for execution.
- It executes much less frequently; minutes may separate the creation of one new process and the next.
- It controls the degree of multiprogramming.

➢ **Short-term Scheduler:**

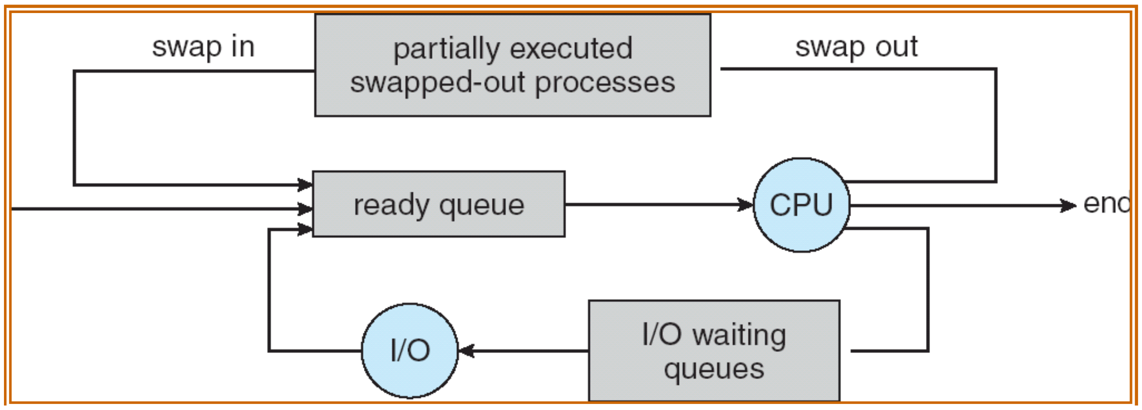
- It is also known as CPU scheduler.
- It selects from among the processes that are ready to execute and allocates the CPU to one of them.
- It must select a new process for the CPU frequently.
- It executes at least once every 100ms.Because of the short time between executions; the short-term scheduler must be fast. If it takes 10ms to

decide to execute a process for 100ms, then 10/(100+10)=9% of the CPU is being used(Wasted) simply for the scheduling work.

- **Medium-term Scheduler:**
  - The medium-term scheduler temporarily removes processes from main memory and places them in secondary memory and vice versa.
  - This is referred as "swapping out" or "swapping in" / "paging out" or "paging in".
  - The medium-term scheduler may decide to swap out a process which has not been active for some time, or a process which has a low priority, or a process which is page faulting frequently, or a process which is taking up a large amount of memory in order to free up main memory for other processes, swapping the process back in later when more memory is available, or when the process has been unblocked and is no longer waiting for a resource.

Processes can be described as either:

- ***I/O-bound process*** – spends more time doing I/O than computations, many short CPU bursts
- ***CPU-bound process*** – spends more time doing computations; few very long CPU bursts



- **Fig: Addition of medium-term scheduling to the queuing diagram**

2.5. Context Switch:

- When CPU switches to another process, the system must save the state of the old process and load the saved state for the new process.
- This is called Context Switch.
- Context-switch time is overhead; the system does no useful work while switching Time dependent on hardware support.
- When a context switch occur kernel saves the context of old process in its PCB and loads the context of new process that is scheduled to run.

➢ **CPU Scheduler**

- Selects from among the processes in memory that are ready to execute, and allocates the CPU to one of them
- CPU scheduling decisions may take place when a process:
  - ➢ Switches from running to waiting state
  - ➢ Switches from running to ready state
  - ➢ Switches from waiting to ready
  - ➢ Terminates
- Scheduling under 1 and 4 is *no preemptive*
- All other scheduling is *preemptive*

2.6. Scheduling Criteria:

- Different CPU Scheduling algorithms have different properties.
- The criteria include the following.

➢ **CPU utilization**: To keep the CPU as busy as possible. it range from 0 to 100 percent. In a real system it should range from 40%(for lightly loaded system) to 90% (for heavily loaded system)

➢ **Throughput**: The number of processes that complete their execution per time unit

➢ **Turnaround time**: The interval from the time of submission of process to the time of completion is the turnaround time. This time is the sum of the periods spent waiting to get into memory, waiting in the ready queue, executing in the CPU and doing I/O. The amount of time to execute a particular process.

- **Waiting time:** The amount of time that a process has been waiting in the ready queue. It is the sum of the periods spent waiting in the ready queue.
- **Response time** : amount of time it takes from when a request was submitted until the first response is produced, not output (for time-sharing environment)

It is desirable to maximize CPU utilization and throughput and to minimize turnaround time, waiting time and response time.

2.7. Scheduling Algorithms :

- CPU scheduling deals with the problem of deciding which of the processes in the ready queue is to be allocated the CPU.
- There are many different CPU scheduling algorithms are there:

1. **FCFS(First –Come, First-Served)**
2. **SJF(Shortest-Job-First)**
3. **Priority Scheduling**
4. **Round-Robin(RR)**

2.7.1 FCFS (First –Come, First-Served):

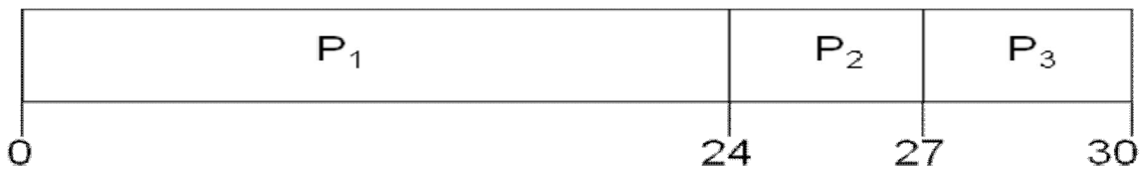
- It is the simplest CPU scheduling algorithm, the process that requests the CPU first is allocated the CPU first.
- The implementation of FCFS is easily managed with FIFO queue.
- When a process enters the ready queue its PCB is linked on to the tail of the queue.
- When CPU is free, it is allocated to the process at the head of the queue.

- FCFS is non preemptive. Once the CPU has been allocated to a process, that process keeps the CPU until it releases the CPU, either by terminating or by requesting I/O.

Example: Consider the following set of processes that arrive at time 0, with the length of the CPU burst given in milliseconds:

Process	**Burst Time**
P_1	**24**
P_2	**3**
P_3	**3**

- Suppose that the processes arrive in the order: P_1 , P_2 , P_3
- The Gantt Chart for the schedule is:

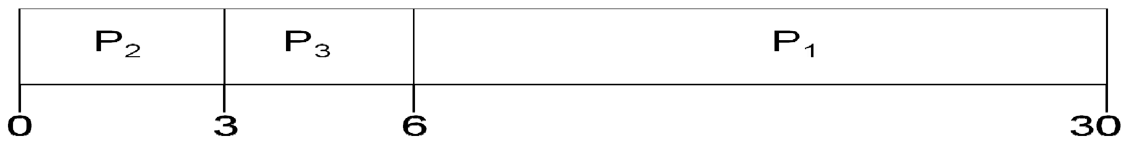


- Waiting time for P_1 = 0; P_2 = 24; P_3 = 27
- Average waiting time: (0 + 24 + 27)/3 = 17
- Suppose that the processes arrive in the order
  - P_2 , P_3 , P_1

Convoy Effect:

- All other processes wait for the one big process to get off the CPU. This effect results in lower CPU utilization and device utilization.

The Gantt chart for the schedule is:



- Waiting time for P_1 = 6; P_2 = 0; P_3 = 3
- Average waiting time: (6 + 0 + 3)/3 = 3
- Much better than previous case

- *Convoy effect* short process behind long process

2.7.2 SJF (Shortest-Job-First):

- Associate with each process the length of its next CPU burst. Use these lengths to schedule the process with the shortest time

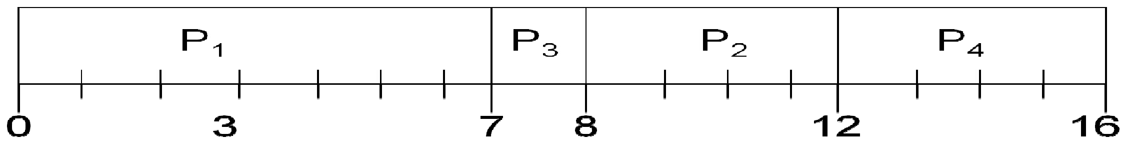
➢ <u>Two schemes:</u>

- **Non preemptive** – once CPU given to the process it cannot be preempted until completes its CPU burst
- **Preemptive** – if a new process arrives with CPU burst length less than remaining time of current executing process, preempt. This scheme is known as the Shortest-Remaining-Time-First (SRTF)
- SJF is optimal – gives minimum average waiting time for a given set of processes

<u>Example of Non-Preemptive SJF</u>

Process	Arrival Time	Burst Time
P_1	0.0	7
P_2	2.0	4
P_3	4.0	1
P_4	5.0	4

- SJF (non-preemptive)

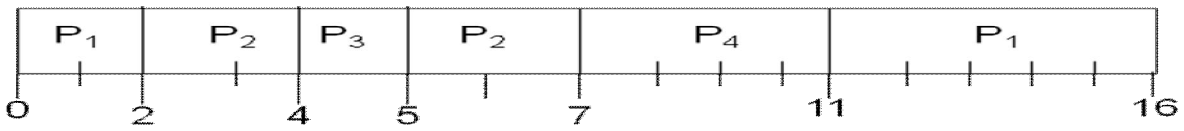


- Average waiting time = (0 + 6 + 3 + 7)/4 = 4

<u>Example of Preemptive SJF:</u>

Process	Arrival Time	Burst Time
P_1	0.0	7
P_2	2.0	4
P_3	4.0	1
P_4	5.0	4

SJF (preemptive)



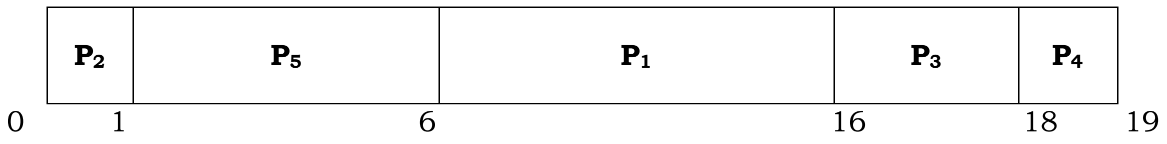
- Average waiting time = (9 + 1 + 0 +2)/4 = 3

2.7.3 Priority Scheduling:

- SJF algorithm is a special case of the general priority scheduling algorithm.
- A priority is associated with each process, and the CPU is allocated to the process with highest priority.
- Equal-priority processes are scheduled in FCFS order.
- An SJF algorithm is simply a priority algorithm where the priority(p) is the inverse of the (predicted) next CPU burst .
- The large CPU burst, the lower the priority, and vice versa.
    - Priorities are generally indicated by some fixed range of numbers, such as 0 to 7, or 0 to 4095. Here we assume low numbers represent high priority.
    - As an example, consider the following set of processes, assumed to have arrived at time 0, in the order p1, p2,…., p5, with the length of the CPU burst given in milliseconds.

Process	**Burst Time**	**Priority**
P_1	**10**	**3**
P_2	**1**	**1**
P_3	**2**	**4**
P_4	**1**	**5**
P_5	**5**	**2**

Using priority scheduling, we would schedule these processes according to the following Gantt chart:



The average waiting time is 8.2 milliseconds.

- Priorities can be defined either internally or externally. Internally defined priorities are influenced by time limits, memory requirements, the number of open files, and the ratio of I/O burst to average CPU burst.
- External priorities are set by criteria outside operating system, such as importance of the process, type and amount of funds being paid for computer use.
- Priority scheduling can be either preemptive or non preemptive.
- When a process arrives at the ready queue, its priority compared with the priority of currently running process.
- A preemptive priority scheduling algorithm will preempt the CPU if the priority of newly arrived process is higher, than the currently running process.
- A non preemptive scheduling algorithm will simply put the new process at the head of the ready queue.

Drawback:

Starvation

- Starvation or indefinite blocking is the major problem in priority scheduling algorithms.
- A process that is ready to run but waiting for the CPU can be considered blocked.
- A priority scheduling algorithm can leave some low priority processes indefinitely.
- High priority processes can prevent low- priority processes from ever getting the CPU.

Aging

- Aging is a solution to the problem of indefinite blocking of low – priority processes.
- Aging is a technique of gradually increasing the priority of a process that wait in the system for a long time.

- **For ex:** Priorities range from low to high that is 127 to 0, we could increase the priority of a waiting process by 1 every 15 minutes.

2.7.4 Round Robin Scheduling:

- This algorithm is designed especially for time sharing systems.
- It is similar to FCFS scheduling, but pre-emption is added to switch between processes.
- A small unit of time called time quantum or time slice is defined.
- Time quantum is generally from 10 to 100 ms. 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 upto 1 time quantum.
- The ready queue acts as FIFO queue of processes, new processes are added to tail of ready queue.
- The CPU scheduler pick the first processes from 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 process.
- In RR scheduling we are having **two possibilities** first one is the process may have CPU burst of less than 1 time quantum.
  - In this case the process itself releases the CPU voluntarily. Scheduler will then proceed to the next process in ready queue.
- Second possibility is CPU burst of currently running process is longer than 1 time quantum, the timer will go off and cause interrupt to the operating system.
- A context switch will be executed, and the process will be executed, and the process will be put any the tail of ready queue.
- The CPU scheduler will then select the next process in the ready queue.
- The average waiting time in RR policy is long.
  - ➢ consider the following set of processes that arrive at 0, with the length of the CPU burst given in milliseconds.

Process	**Burst Time**
P_1	**24**
P_2	**3**
P_3	**3**

- ➢ If we use 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 CPU is given to the next process in the queue, process p2.
- ➢ Since p2 does not need 4 milliseconds, it quits before its time quantum expires. The CPU is given to process p3.
- ➢ Once each process received 1 time quantum the CPU is returned to process p1 for an additional time quantum

P1	**P2**	**P3**	**P1**	**P1**	**P1**	**P1**	**P1**

0 4 7 10 14 18 22 26 30

The average waiting time is 17/3=5.66 ms

2.7.5 Multilevel Queue Scheduling:

- Processes are easily classified into different groups.
- For example common division is made between foreground (interactive) processes and background (batch) processes.
- These two types of processes have different response time requirements.

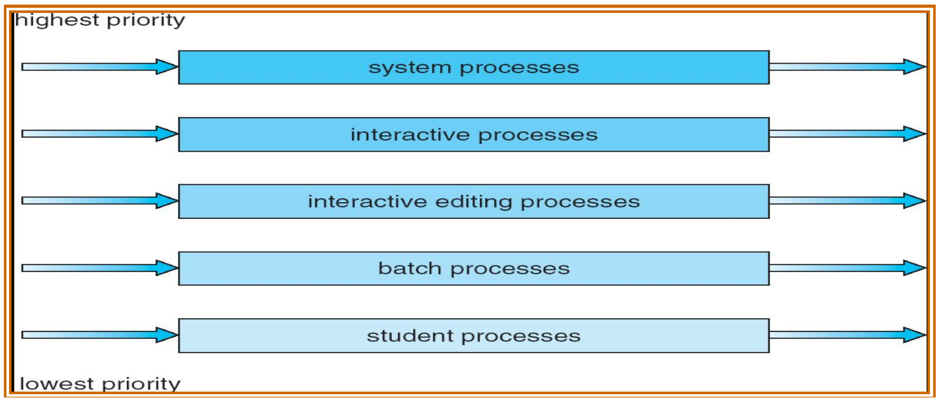


Fig: Multilevel queue scheduling.

- A multi level queue scheduling algorithm partitions ready queue into several separate queues.
- The processes are permanently assigned to one queue, generally based on some property of process, such as memory size, process priority, or process type.
- For example, separate queues might be used for foreground and background processes.
- The foreground queue might be scheduled by an RR- algorithm.
- Example of a multilevel queue scheduling algorithm with five queues, listed below in order of priority.

1. System processes
2. Interactive process
3. Interactive editing processes
4. Batch processes
5. Student processes

- If interactive editing processes entered the ready queue while a batch processes was running, the batch process would be pre-empted.
- The foreground queue can be given can be given 80 percent of CPU time for RR scheduling among its processes, where as the background queue receives 20 percent of CPU time for its processes in FCFS basis.

2.7.6 Multilevel Feedback Queue Scheduling:

- This algorithm allows a process to move between queues. The idea is to separate processes according to the CPU bursts.
- If a process uses too much CPU time, it will be moved to lower-priority queue.
- A process that waits too long in a lower priority queue may be moved to a higher priority queue.
- For example, consider a multilevel feedback queue scheduler with three queues, numbered from 0 to 2. The scheduler first executes all the process in the queue 0.

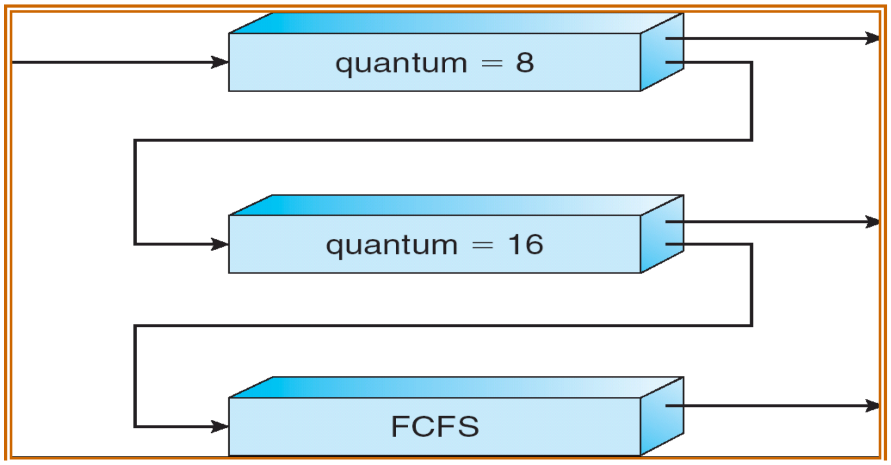


Fig: Multilevel feedback queues.

- Only when queue 0 is empty it executes processes in queue 1.
- Processes in queue 2 will only be executed if queue 0 and 1 are empty.
- A process that arrives for queue 1 will in turn be pre-empted by a process arriving for queue 0.

2.8. Operations on Processes

- Process in most systems can execute concurrently and they may be created and deleted dynamically.
    - ➢ Process Creation
    - ➢ Process Termination

2.8.1 Process Creation:

- A process may create several new processes by using Create-process system call, during execution.
- Parent process creates children processes, which, in turn create other processes, forming a tree of processes.
- In most operating system s a process can be identified by using a unique process identifier (pid), which is typically an integer number.

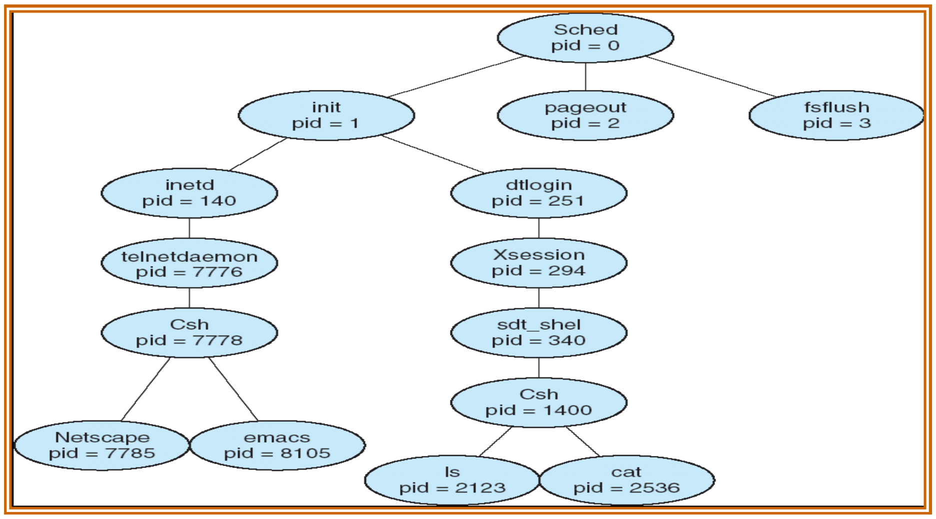


Fig: A tree of processes on a typical Solaris system

When a process creates a new process, two possibilities exist in terms of execution:

- ➢ Parent and children execute concurrently
- ➢ Parent waits until children terminate

Resource sharing

- ➢ Parent and children share all resources
- ➢ Children share subset of parent's resources
- ➢ Parent and child share no resources

- A new process is created by the fork ( ) system call.
- The new process consists of copy of Address space of the original process. Here parent process easily communicates with its child process.
- Both parent and child process continues execution at the instruction after the fork ( ) with one difference:
  - The return code for fork ( ) is zero for child process whereas the non zero process identifier of the child is returned to parent.

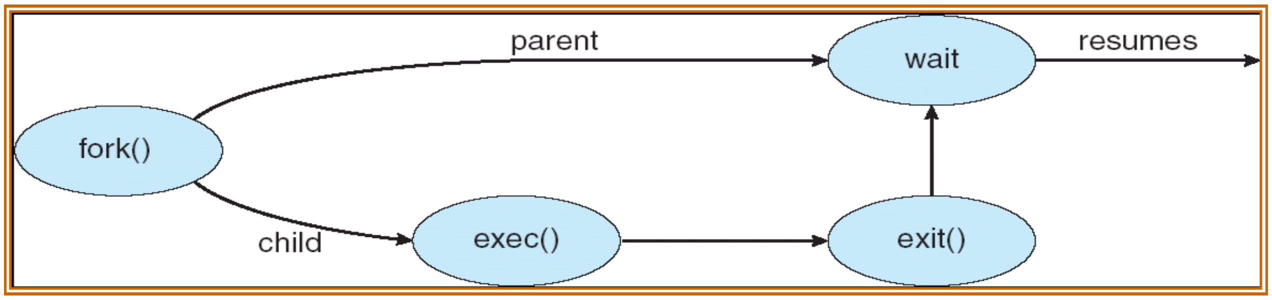


Fig: Process Creation

- The exec ( ) system call is invoked after fork ( ) and then parent will create more children or if it has nothing else to do it waits while the child runs by invoking wait ( ) system call.
- If child completes its execution and terminates then parent also terminates using exit ( ) system call.

2.8.2 Process Termination:

- A Process terminates when it finishes executing its last statement and asks the operating system to delete it by using exit ( ) system call.
- All the resources the resources of the process including physical and virtual memory, open files and I/O buffers and deallocated by the operating system.
- Termination can occur in other circumstances as the process can cause termination of another process. Parent process can terminate its child process.

  Parent may terminate execution of children processes (abort)
  - ➢ Child has exceeded allocated resources
  - ➢ Task assigned to child is no longer required
  - ➢ If parent is exiting
- Some operating system do not allow child to continue if its parent terminates.
- If a process terminates either normally or abnormally, then all of its children must also be terminated.

- This phenomenon is referred to as - cascading termination

2.9. Inter-process Communication:

- Processes executing concurrently in the operating system may be either independent process or cooperating process.
  - **Independent Process:**
    An independent process is a process which cannot affect or be affected by other processes in the system. Any process that does not share data with any other process is independent.
  - **Cooperating Process:**
    It can effect or be affected by other process executing in the system. Any process that shares information with other process is a cooperating process.

There are several reasons behind process cooperation:

1. **Information Sharing:** Several users may be interested in the same piece of information for example a shared file; we must allow concurrent access to such information.
2. **Computation Speed:** If we want a particular task to run faster, we must break it into subtasks, each of which will be executing in parallel with others.
3. **Modularity:** We may want to construct the system in a modular fashion, dividing the system functions into separate processes or threads.
4. **Convenience:** Even an individual user may work on many tasks at the same time. For instance, a user may be editing, printing and compiling in parallel.

- Cooperating processes require an inter process communication (IPC), which allow them to exchange data and information.

There are two fundamental models in Inter Process Communication:

1. Shared memory systems.
2. Message Passing Systems.

- In shared- memory model, a region of memory that is shared by cooperating processes is established.
- Processes can then exchange information by reading and writing the data to the shared region.
- In message- passing, the communication takes place by means of messages exchanged between cooperating processes.

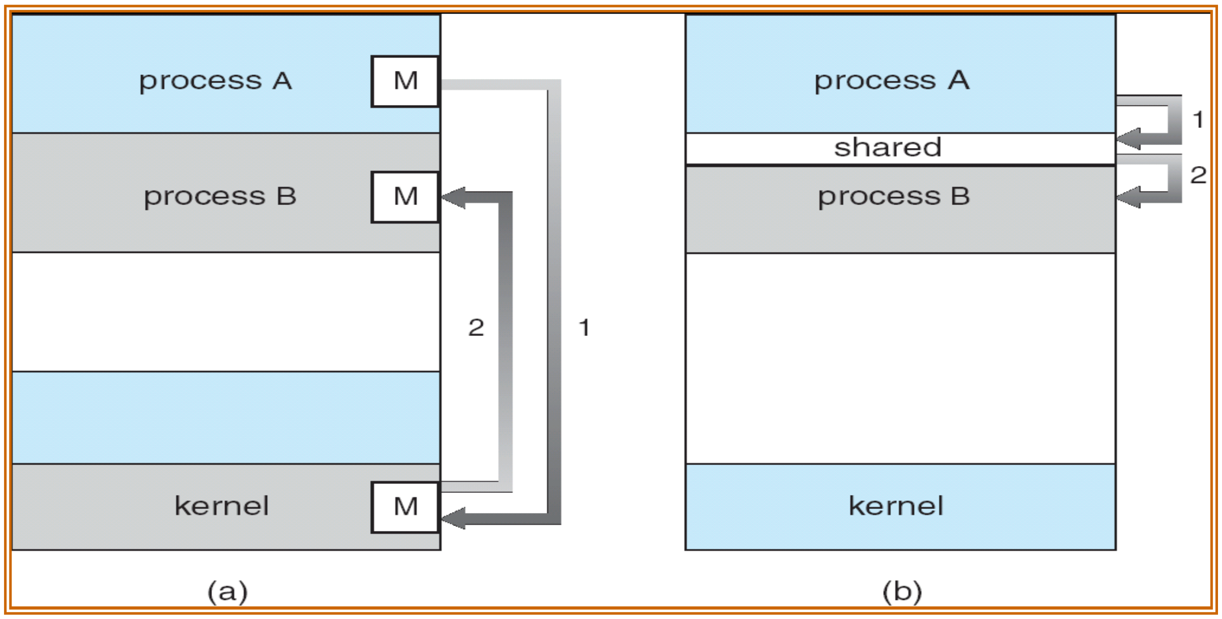


Fig: Communications models (a) Message passing (b) Shared Memory

- Message passing is useful for exchanging smaller amounts of data, it is also easier to implement than shared memory.
- Shared memory allows maximum speed and convenience of communication, as it can be done at memory speeds when within a computer.
- Shared memory is faster than message passing systems.

2.9.1 Shared memory Systems:

- Inter-process communication using shared memory requires communicating processes to establish a region of shared memory.
- A shared memory region resides in the address space of the process creating the shared memory segment.
- The other processes that wish to communicate using this shared memory segment must attach it to their address space.
- Ex: The concept of cooperating process can be illustrated by Producer-Consumer problem.
- The producer produces the information that is consumed by a consumer process.
- If producer and consumer process are executing concurrently, then the buffer is filled by producer and emptied by consumer.

  Two types of buffers are used:

  Unbounded Buffer: This has no limit on the size of buffer. The consumer may have to wait for new items, but producer can always produce new items.

  Bounded Buffer: Here buffer size is fixed. The consumer must wait, if buffer is empty and producer must wait if buffer is full.

2.9.2 Message Passing System:

- This system allows processes to communicate without sharing some address space, here communicating processes may reside on different computers connected by a network.
- A message passing facility provides at least two operations: send (message) and receive (message).
- Messages sent by a process can be either fixed size or variable size.
- ***Fixed size:*** If fixed size messages only sent, the system level implementation is straight forward and programming is more difficult.

- ***Variable size:*** Variable sized messages required a more complex system level implementation, but programming task become simpler.
- If process p and q want to communicate they must send messages to and receive messages from each other, a communication link must exist between them.
- The link may be physical implementation such as shared memory hardware bus, or network or logical implementation.
- Logical implementation of a link include send ( )/ receive ( ) operations.
  1. Naming
  2. Synchronization
  3. Buffering

1. Naming:

- Processes that want to communicate must have a way to refer each other. They use either direct or indirect communication.

Direct Communication

- Each process that wants to communicate must explicitly name the recipient or sender of the communication.

➢ Send (p, message) ---- send a message to process p.

➢ Receive (q, message) --- receive a message from process q.

- Here communication link is established with exactly two processes.
- This scheme exhibits symmetry in addressing;
- Therefore both sender and receiver processes must name the other process. Asymmetric in addressing, here only sender names, the recipient, where recipient is not require to name the sender.

➢ Send (p, message) ---- send a message to process p.

➢ Receive (id, message) --- receive a message from any process.

- The variable id is set to the name of process with which communication has taken place.

Indirect Communication

- Here, the messages are sent to and received from mailboxes or ports.
- A process can communicate with other process can communicate with other process only if the processes has shared mailbox.

➢ Send (A, message) ---- send a message to mailbox A.

➢ Receive (A, message) --- receive a message from mailbox A.

- Here communication link may be associated with more than two processes.
- For example the processes p1, p2 and p3 all shared mailbox A. Process P1 sends message to A. while both P2 and P3 both executes receive() from A.
- The operating system will provide a mechanism that allows a process to do the following:

➢ Create a new mail box

➢ Send and receive messages through the mail box

➢ Delete mail box

2. <u>Synchronization</u>

- Communication between processes takes place through calls to send () and receive ( ) primitives.
- Message passing may be either blocking or non blocking also known as synchronous and non synchronous.
- **Blocking send:** The sending process is blocked until the message is received by receiving process or by the mailbox.
- **Non-Blocking send**: The sending process sends the message and resumes operation.
- **Blocking receive:** The receiver blocks until a message is available
- **Non Blocking receiver:** The receiver retrieves either a valid message or null.

3. Buffering

- Whether a communication is direct or indirect, messages exchanged by communicating processes reside in temporary queue.

These queues are 3 types.

- **Zero capacit**y: Maximum length of queue is zero, here link cannot have any messages waiting in it, therefore the sender must block until recipient receives message.
- **Bounded Capacity**: Queue has finite length n; thus almost n messages can reside in it. If queue is not full when a new message is sent, the message is placed in queue, and sender continues execution without waiting. If link is full the sender must block until space is available.

****

UNIT-II
Assignment-Cum-Tutorial Questions
SECTION-A

Objective Questions

1. Long term scheduler is also known as______________ [ ]
   A. High scheduler C. CPU scheduler
   B. Job scheduler D. None of the above
2. Short term scheduler is also known as______________ [ ]
   A. High scheduler C. CPU scheduler
   B. Job scheduler D. None of the above
3. Process Control Block is also called as________________ [ ]
   A. Program Control Block C. Project Control Block
   B. Task Control Block D. Procedure Control Block
4. In which of the following scheduling policies does context switching never takes place. [ ]
   A. Round-Robin C. Shortest Job First
   B. Preemptive D. FCFS
5. Suppose that a process is in 'BLOCKED' state waiting for some I/O service. When the service is completed, it goes to the [ ]
   A. Running State C. Ready State
   B. Suspended State D. Terminated State
6. Which of the following scheduling algorithm give minimum average waiting time? [ ]
   A. FCFS B. SJF C. Round-Robin D. Priority
7. Which of the following scheduling policy is well suited for time-shared OS
   A. FCFS B. SJF C. Round-Robin D. Elevator [ ]
8. Shortest Remaining Time First is a preemptive version of ____ [ ]
   A. FCFS B. SJF C. Round-Robin D. Priority
9. Which combination of the following features will suffice to characterize an OS as a multi-programmed OS? **(GATE-2002)**

I. More than one program may be loaded into main memory at the same time for execution.

II. If a program waits for certain events such as I/O, another program is immediately scheduled for execution.

III. If the execution of a program terminates, another program is immediately scheduled for execution. [ ]

A. i B. i and ii C. i and iii D. i, ii and iii

10. The processes that are residing in main memory and are ready and waiting to execute are kept on a list called __________ [ ]

A. Job queue B. Ready queue C. Device queue D. FIFO queue

11. Which of the following statement(s) is false about SJF? [ ]

S1: It causes minimum average waiting time

S2: It can cause starvation

A. Only S1 B. Only S2 C. Both S1 and S2 D. Neither S1 nor S2

12. Pre-emptive scheduling is the strategy of temporarily suspending a running process [ ]

A. before the CPU time slice expires

B. to allow starving processes to run

C. when it requests I/O

D. to avoid collision

13. What is the range of a time quantum in Round-Robin Scheduling?

A. 10-100 ms C. 10-100 ns [ ]

B. 100-1000 ms D. 100-1000ns

14. As a rule of thumb what percentage of the CPU bursts should be shorter than the time quantum? [ ]

A. 80% B. 70% C. 60% D. 50%

15. Interval between the time since submission of the job to the time its results become available, is called [ ]

A. Response Time C. Throughput

B. Waiting time D. Turnaround Time

16. The scheduling in which CPU is allocated to the process with least CPU-burst time is called [ ]

A. Priority Scheduling C. Round Robin Scheduling

B. Multilevel Queue Scheduling D. Shortest job first Scheduling

17. Which scheduling policy is used for a batch processing operating system

A. Shortest-job First. C. Round-Robin. [ ]

B. Priority Based D. First-Come-First-Serve.

18. Which of these is a technique of improving the priority of process waiting in Queue for CPU allocation [ ]

A. Starvation B. Relocation C. Promotion D. Aging

19. Consider a set of n tasks with known runtimes r1, r2, … rn to be run on a uniprocessor machine. Which of the following processor scheduling algorithms will result in the maximum throughput? **(GATE-2001)**

A. Round-Robin C. Shortest-Job-First [ ]

B. Highest-Response-Ratio-Next D. First-Come-First-Served

20. Which of the following scheduling algorithms is non-preemptive? (**GATE CS 2002**)

A. Round Robin [ ]

B. First come first serve

C. Multilevel Queue Scheduling

D. Multilevel Queue Scheduling with Feedback

SECTION-B

Descriptive Questions

1. With a neat sketch explain **process state diagram**?
2. Explain about the contents of **process control block**?
3. Define long term scheduler and short term scheduler?
4. Compare and contrast short term, medium term and long term scheduling.?
5. Discuss **criteria** involved in scheduling a process?

6. Explain about inter process communication **(IPC)**?
7. Demonstrate two different **operations** performed on processes?
8. What is convey effect? Explain with an example?
9. Discuss the **problem** involved in priority scheduling algorithm with a suitable example and provide a **solution** to that problem?
10. Differentiate shared memory and message passing models of process communication?
11. Explain the role of schedulers with the help of process transition diagram?
12. With a suitable example explain about context switching?
13. Write about Priority and SJF(Shortest Job First) scheduling algorithms with an example.

Problems:

1. Suppose that the following processes arrive for execution at the times indicated

Process	Arrival Time	Burst Time
P_1	0.0	8
P_2	0.4	4
P_3	1.0	1

What is the average waiting and turnaround time for these processes using

a) FCFS scheduling algorithm

b) SJF Non Preemptive scheduling algorithm

c) SJF Preemptive scheduling algorithm

2. Consider the following processes, with the arrival time and the length of the CPU burst given in milliseconds.

Process	Arrival Time	Burst Time
P_1	0	10
P_2	3	6
P_3	7	1
P_4	8	3

Calculate average waiting and average turnaround time using

a) Non preemptive priority CPU scheduling algorithm

b) Preemptive priority CPU scheduling algorithm

c) Round robin scheduling algorithm(TQ=3ms)

3. Consider the following set of processes, with the arrival times and the CPU-burst times given in milliseconds (**GATE-CS-2004**)

Process	**Arrival Time**	**Burst Time**
P1	0	5
P2	1	3
P3	2	3
P4	4	1

What is the average turnaround time for these processes with the preemptive shortest remaining processing time first (SRTF) algorithm ?

4. Consider the following set of Processes with CPU Burst times in milliseconds, arrival times in milliseconds and Priorities:

Process	**Burst time**	**Arrival Time**	**Priority**
P1	8	1	2
P2	5	0	1
P3	14	2	4
P4	3	4	3

Draw the Gantt Chart. Calculate Average Turnaround Time and Average Waiting Time by using:

i) Round Robin (if Time Quantum = 4msec)

ii) Priority Scheduling.(both preemption and non preemption)

SECTION-C

QUESTIONS AT THE LEVEL OF GATE

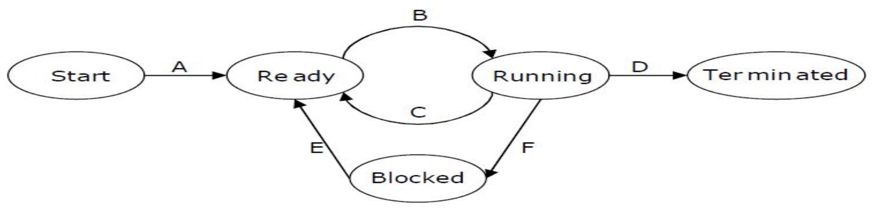
1. An operating system uses Shortest Remaining Time first (SRT) process scheduling algorithm. Consider the arrival times and execution times for the following processes: **[GATE 2007]**

Process	Execution time	Arrival time
P1	20	0
P2	25	15
P3	10	30
P4	15	45

What is the total waiting time for process P2? [ ]

(A) 5 (B)15 (C)40 (D)55

2. In the following process state transition diagram for a uni processor system, assume that there are always some processes in the ready state: Now consider the following statements: **[GATE 2009]**



I. If a process makes a transition D, it would result in another process making transition A immediately.

II. A process P2 in blocked state can make transition E while another process P1 is in running state.

III. The OS uses pre emptive scheduling.

IV. The OS uses non-pre emptive scheduling.

Which of the above statements are TRUE? [ ]

A. I and II B. I and III C. II and III D. II and IV

3. Which of the following statements are true? **[GATE 2010]**

a) Shortest remaining time first scheduling may cause starvation

b) Pre emptive scheduling may cause starvation

c) Round robin is better than FCFS in terms of response time [ ]

A. I only B. II and III only C. I and III only D. I,II and III.

4. Consider the following table of arrival time and burst time for three processes P0, P1 and P2. **[GATE 2011]**

Process	Arrival time	Burst Time
P0	0 ms	9 ms
P1	1 ms	4 ms
P2	2 ms	9 ms

The pre-emptive shortest job first scheduling algorithm is used. Scheduling is carried out only at arrival or completion of processes. What is the average waiting time for the three processes?

A. 5.0 ms B. 4.33 ms C. 6.33 D. 7.33.

5. Consider the 3 processes, P1, P2 and P3 shown in the table. **[GATE 2012]**

Process	Arrival time	Time Units Required
P1	0	5
P2	1	7
P3	3	4

The completion order of the 3 processes under the policies FCFS and RR2 (round robin scheduling with CPU quantum of 2 time units) are [ ]

A. FCFS: P1, P2, P3
RR2: P1, P2, P3

B. FCFS: P1, P3, P2
RR2: P1, P3, P2

C. FCFS: P1, P2, P3
RR2: P1, P3, P2

D. FCFS: P1, P3, P2
RR2: P1, P2, P3

6. A scheduling algorithm assigns priority proportional to the waiting time of a process. Every process starts with priority zero (the lowest priority). The scheduler re-evaluates the process priorities every T time units and decides the next process to schedule. Which one of the following is TRUE if the processes have no I/O operations and all arrive at time zero?

[GATE2013]

A. This algorithm is equivalent to the first-come-first-serve algorithm.[ ]

B. This algorithm is equivalent to the round-robin algorithm.

C. This algorithm is equivalent to the shortest-job-first algorithm.

D. This algorithm is equivalent to the shortest-remaining-time-first algorithm.

7. An operating system uses *shortest remaining time first* scheduling algorithm for pre-emptive scheduling of processes. Consider the following set of processes with their arrival times and CPU burst times (in milliseconds):

Process	Arrival time	Burst Time
P1	0	12
P2	2	4
P3	3	6
P4	8	5

The average waiting time (in milliseconds) of the processes is __[**GATE-2014]**

8. Consider the following set of processes that need to be scheduled on a single CPU. All the times are given in milliseconds.

Process Name	Arrival Time	Execution Time
A	0	6
B	3	2
C	5	4
D	7	6

E	10	3

Using the *shortest remaining time first* scheduling algorithm, the average process turnaround time (in msec) is **[GATE-2014]**

9. Consider a uniprocessor system executing three tasks T_1, T_2 and T_3, each of which is composed of an infinite sequence of jobs (or instances) which arrive periodically at intervals of 3, 7 and 20 milliseconds, respectively. The priority of each task is the inverse of its period, and the available tasks are scheduled in order of priority, with the highest priority task scheduled first. Each instance of T_1, $T_2$and T_3 requires an execution time of 1, 2 and 4 milliseconds, respectively. Given that all tasks initially arrive at the beginning of the 1stmillisecond and task preemptions are allowed, the first instance of T_3 completes its execution at the end of ______ milliseconds.

[GATE-2015]

A. 5 B. 10 C. 12 D.15 [ ]

10. For the processes listed in the following table, which of the following scheduling schemes will give the lowest average turnaround time?

[GATE-2015]

Process	Arrival Time	Burst Time
A	0	3
B	1	6
C	4	4
D	6	2

A. First Come First Serve [ ]
B. Non – preemptive Shortest Job First
C. Shortest Remaining Time
D. Round Robin with Quantum value two

11. Consider the following processes, with the arrival time and the length of the CPU burst given in milliseconds. The scheduling algorithm used is preemptive shortest remaining-time first.

Process	**Arrival Time**	**Burst Time**
P1	0	10
P2	3	6
P3	7	1
P4	8	3

The average turnaround time of these processes is_____. **[GATE-2016]**

12. Consider the following CPU processes with arrival times (in milli seconds) and length of CPU bursts (in milli seconds) as given below: **[GATE-2017]**

Process	Arrival time	Burst time
P1	0	7
P2	3	3
P3	5	5
P4	6	2

If the pre-emptive shortest remaining time first scheduling algorithm is used to schedule the processes, then the average waiting time across all processes is_______________ milliseconds.

13. Consider the set of processes with arrival time (in milliseconds), CPU burst time (in milliseconds) , and priority (0 is the highest priority) shown below. None of the processes have I/O burst time.

Process	Arrival time	Burst Time	Priority
P1	0	11	2
P2	5	28	0
P3	12	2	3

P4	2	10	1
P5	9	16	4

The average waiting time (in milliseconds) of all the processes using preemptive priority scheduling algorithm is_________ **[GATE-2017]**

14. Consider the following four processes with arrival times (in milliseconds) and their length of CPU burst (in milliseconds) as shown below: **[GATE-2019]**

Process	P1	P2	P3	P4
Arrival time	0	1	3	4
CPU burst time	3	1	3	Z

These processes are run on a single processor using preemptive Shortest Remaining Time First scheduling algorithm. If the average waiting time of the processes is 1 millisecond, then the value of Z is __________.

(A) 2

(B) 3

(C) 1

(D) 4