Chapter 2

Processes and Process Management

Amol D. Vibhute (PhD)
Assistant Professor

Email ID:- amol.vibhute@sicsr.ac.in

Roadmap of Chapter

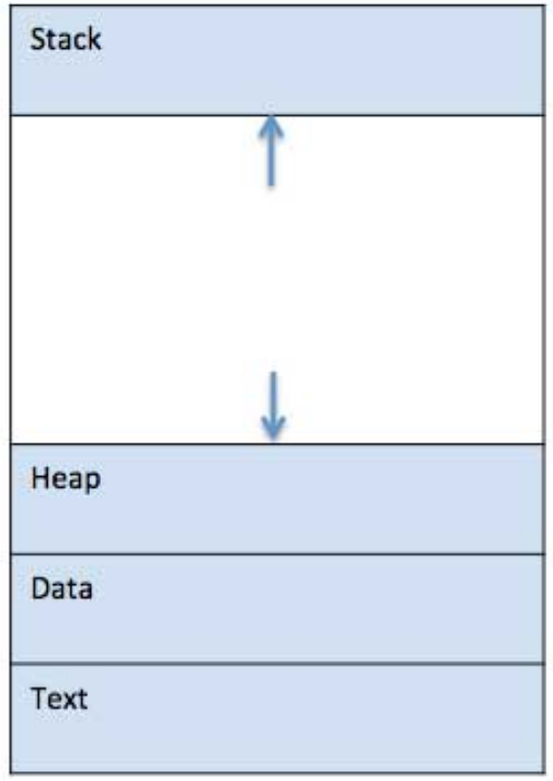
- Processes and Process Management
  - Process concept and process states,
  - CPU and I/O bound,
  - Operating system services for process and thread management,
  - CPU scheduler- short, medium, long-term, dispatcher,
  - Scheduling: - preemptive and non-preemptive
  - Scheduling algorithms- FCFS, SJFS, shortest remaining time, RR, priority scheduling, Multilevel feedback queue.

Introduction:

- A process is the unit of work in modern time-sharing systems.
- A system has a collection of processes – user processes as well as system processes.
- All these processes can execute concurrently with the CPU multiplexed among them.
- A process is a **program in execution (running program)**.
- The execution of a process progresses in a sequential fashion.
- A program is a passive entity while a process is an active entity.
- A process includes much more than just the program code.

Cont....

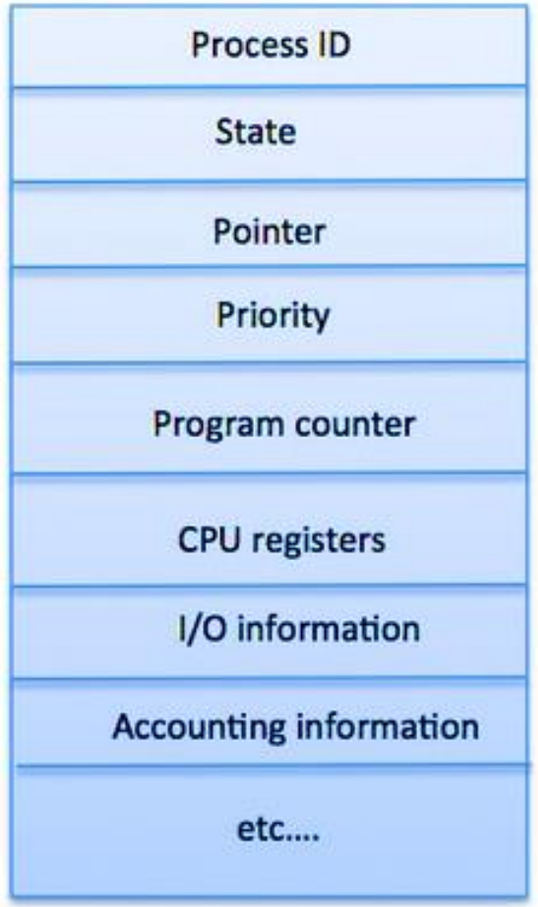
- A process is defined as an entity which represents the basic unit of work to be implemented in the system.
- An instance of running program. The entity that can be assigned to process set execute the operation.
- A computer program is become a process which is in **main memory (RAM)**.
- To put it in simple terms, we write our computer programs in a text file and when we execute this program, it becomes a process which performs all the tasks mentioned in the program.
- When a program is loaded into the memory and it becomes a process, it can be divided into four sections ─ stack, heap, text and data.
- The figure shows a simplified layout of a process inside main memory −



S.N.	Component & Description
1	**Stack:** The process Stack contains the temporary data such as method/function parameters, return address and local variables. The stack is used whenever there is a function call in the program.
2	**Heap:** This is dynamically allocated memory to a process during its run time.
3	**Text:** This includes the current activity represented by the value of Program Counter and the contents of the processor's registers. It consists of the set of instructions to be executed for the process. Also known as code segment.
4	**Data:** This section contains the global and static variables.

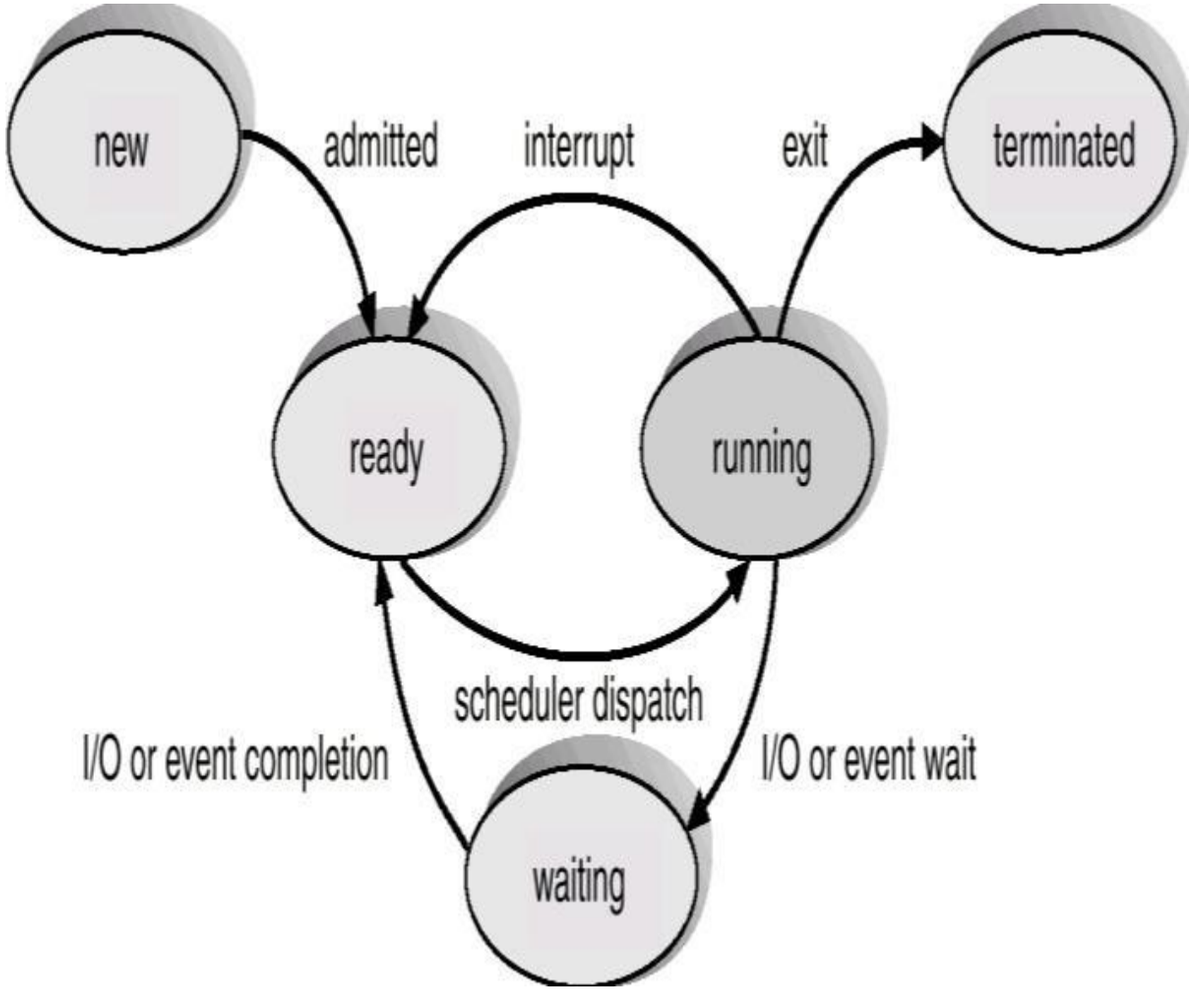
Process Control Block (PCB):

- A Process Control Block is a **data structure maintained by the Operating System for every process**. The PCB is identified by an **integer process ID (PID)**. A PCB keeps all the information needed to keep track of a process also known as **task control block**.
- **Process ID**: Unique identification for each of the process in the operating system.
- **Process State**: The current state of the process i.e., whether it is ready, running, waiting, or whatever.
- **Process privileges**: This is required to allow/disallow access to system resources.
- **Pointer**: A pointer to parent process.
- **Program Counter**: Program Counter is a pointer to the address of the next instruction to be executed for this process.
- **CPU registers**: Various CPU registers where process need to be stored for execution for running state.
- **CPU Scheduling Information**: Process priority and other scheduling information which is required to schedule the process.
- **Memory management information**: This includes the information of page table, memory limits, Segment table depending on memory used by the operating system.
- **Accounting information**: This includes the amount of CPU used for process execution, time limits, execution ID etc.
- **IO status information**: This includes a list of I/O devices allocated to the process.



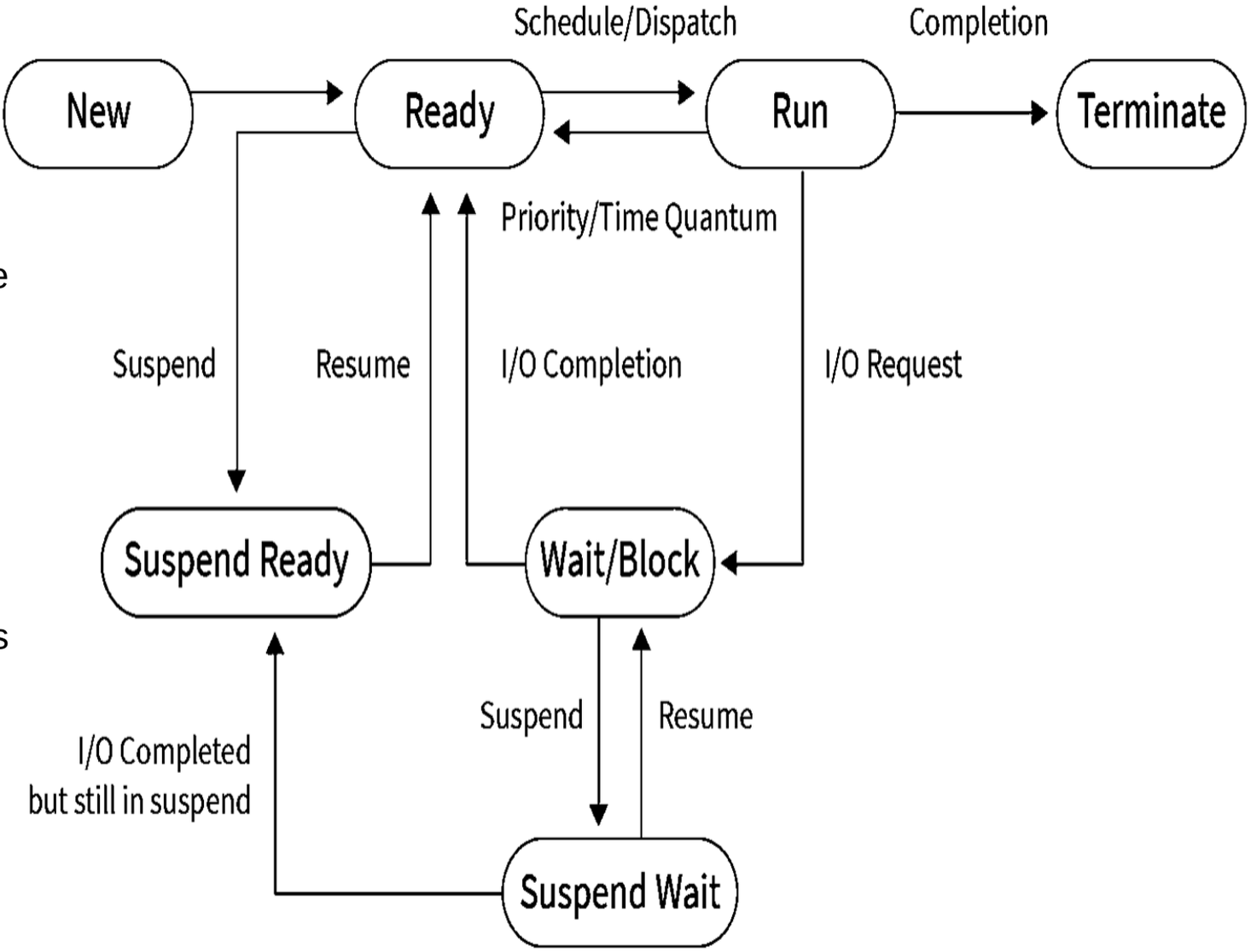
Process States:

- As a process executes, it changes state. The state of a process refers to what the process currently does. A process can be in one of the following states during its lifetime:
  - **New/start**: The process is being created.
  - **Ready**: The process is waiting to be assigned to a processor. Ready processes are waiting to have the processor allocated to them by the operating system so that they can run. Process may come into this state after **Start** state or while running it by but interrupted by the scheduler to assign CPU to some other process.
  - **Running**: Once the process has been assigned to a processor by the OS scheduler, the process state is set to running and the processor executes its instructions.
  - **Waiting**: The process is waiting for some event to occur (such as an I/O completion or reception of a signal).
  - **Terminated or Exit**: Once the process finishes its execution, or it is terminated by the operating system, it is moved to the terminated state where it waits to be removed from main memory.



Cont....

- **Suspend Ready**: Process that was initially in the ready state but was swapped out of main memory(refer to Virtual Memory topic) and placed onto external storage by the scheduler is said to be in suspend ready state. The process will transition back to a ready state whenever the process is again brought onto the main memory.
- **Suspend Wait or Suspend Blocked**: Similar to suspend ready but uses the process which was performing I/O operation and lack of main memory caused them to move to secondary memory. When work is finished it may go to suspend ready.
- **CPU and I/O Bound Processes**: If the process is intensive in terms of CPU operations, then it is called CPU bound process. Similarly, If the process is intensive in terms of I/O operations then it is called I/O bound process.



CPU scheduler:

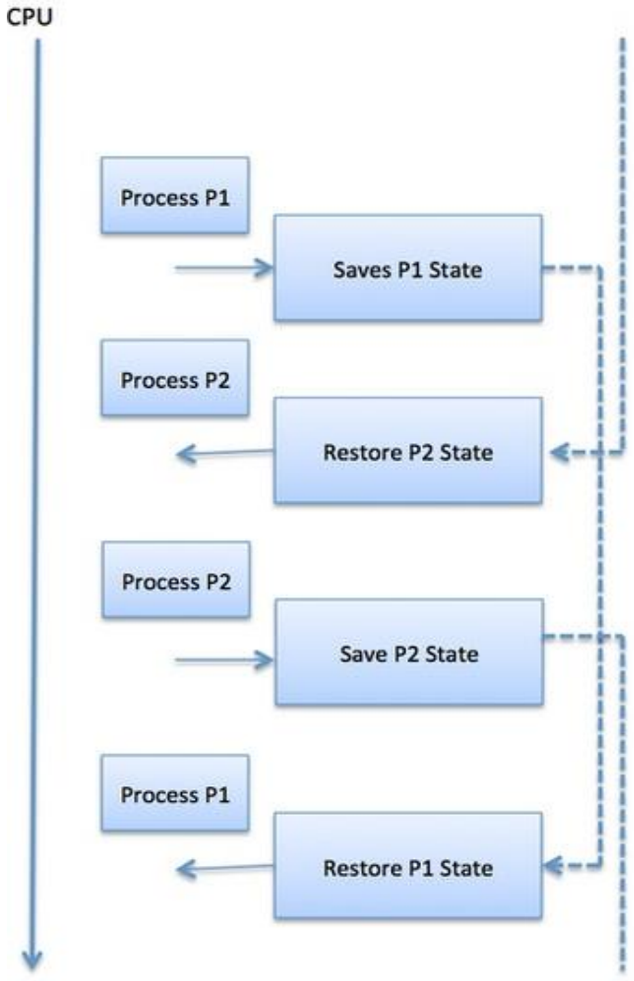
- **Long-Term Scheduler**: Decides how many processes should be made to stay in the ready state. This decides the degree of multiprogramming. Once a decision is taken it lasts for a long time which also indicates that it runs infrequently. Hence it is called a long-term scheduler.
- **Short-Term Scheduler**: Short-term scheduler will decide which process is to be executed next and then it will call the dispatcher. A dispatcher is a software that moves the process from ready to run and vice versa. In other words, it is context switching. It runs frequently. Short-term scheduler is also called CPU scheduler.
- **Medium Scheduler**: Suspension decision is taken by the medium-term scheduler. The medium-term scheduler is used for swapping which is moving the process from main memory to secondary and vice versa. The swapping is done to reduce degree of multiprogramming.

Cont....

S.N.	Long-Term Scheduler	Short-Term Scheduler	Medium-Term Scheduler
1	It is a job scheduler	It is a CPU scheduler	It is a process swapping scheduler.
2	Speed is lesser than short term scheduler	Speed is fastest among other two	Speed is in between both short and long term scheduler.
3	It controls the degree of multiprogramming	It provides lesser control over degree of multiprogramming	It reduces the degree of multiprogramming.
4	It is almost absent or minimal in time sharing system	It is also minimal in time sharing system	It is a part of Time sharing systems.
5	It selects processes from pool and loads them into memory for execution	It selects those processes which are ready to execute	It can re-introduce the process into memory and execution can be continued.

Context Switch:

- A context switch is the mechanism to store and restore the state or context of a CPU in Process Control block so that a process execution can be resumed from the same point at a later time. Using this technique, a context switcher enables multiple processes to share a single CPU. Context switching is an essential part of a multitasking operating system features.
- When the scheduler switches the CPU from executing one process to execute another, the state from the current running process is stored into the process control block. After this, the state for the process to run next is loaded from its own PCB and used to set the PC, registers, etc. At that point, the second process can start executing.
- Context switches are computationally intensive since register and memory state must be saved and restored. To avoid the amount of context switching time, some hardware systems employ two or more sets of processor registers. When the process is switched, the following information is stored for later use.
  - Program Counter
  - Scheduling information
  - Base and limit register value
  - Currently used register
  - Changed State
  - I/O State information
  - Accounting information



Process Scheduling:

- The process scheduling is the activity of the process manager that handles the removal of the running process from the CPU and the selection of another process on the basis of a particular strategy.
- Process scheduling is an essential part of a Multiprogramming operating systems. Such operating systems allow more than one process to be loaded into the executable memory at a time and the loaded process shares the CPU using time multiplexing.
- **Scheduling criteria:** The criteria include the following:
- CPU utilization –
  - The main objective of any CPU scheduling algorithm is to keep the CPU as busy as possible. Theoretically, CPU utilization can range from 0 to 100 but in a real-time system, it varies from 40 to 90 percent depending on the load upon the system.
- Throughput –
  - A measure of the work done by CPU is the number of processes being executed and completed per unit time. This is called throughput. The throughput may vary depending upon the length or duration of processes.
- Turnaround time –
  - For a particular process, an important criteria is how long it takes to execute that process. The time elapsed from the time of submission of a process to the time of completion is known as the turnaround time. Turn-around time is the sum of times spent waiting to get into memory, waiting in ready queue, executing in CPU, and waiting for I/O.
- Waiting time –
  - A scheduling algorithm does not affect the time required to complete the process once it starts execution. It only affects the waiting time of a process i.e. time spent by a process waiting in the ready queue.
- Response time –
  - In an interactive system, turn-around time is not the best criteria. A process may produce some output fairly early and continue computing new results while previous results are being output to the user. Thus another criteria is the time taken from submission of the process of request until the first response is produced. This measure is called response time.

Cont….

- **Scheduling algorithms**:
  - CPU Scheduling is a process of determining which process will own CPU for execution while another process is on hold.
  - A way or selecting a process from ready queue and put it in the CPU.
  - The main task of CPU scheduling is to make sure that whenever the CPU remains idle, the OS at least select one of the processes available in the ready queue for execution.
  - The selection process will be carried out by the CPU scheduler.
  - It selects one of the processes in memory that are ready for execution.
  - **Types of CPU Scheduling**:
    - **Preemptive Scheduling:**
      - In Preemptive Scheduling, the tasks are mostly assigned with their priorities. Sometimes it is important to run a task with a higher priority before another lower priority task, even if the lower priority task is still running. The lower priority task holds for some time and resumes when the higher priority task finishes its execution.
    - **Non-Preemptive Scheduling:**
      - In this type of scheduling method, the CPU has been allocated to a specific process. The process that keeps the CPU busy will release the CPU either by switching context or terminating. It is the only method that can be used for various hardware platforms. That's because it doesn't need special hardware (for example, a timer) like preemptive scheduling.

Cont….

- **When scheduling is Preemptive or Non-Preemptive**?
- To determine if scheduling is preemptive or non-preemptive, consider these four parameters:
  1. A process switches from the running to the waiting state.
  2. Specific process switches from the running state to the ready state.
  3. Specific process switches from the waiting state to the ready state.
  4. Process finished its execution and terminated.
- Only conditions 1 and 4 apply, the scheduling is called non- preemptive.
- All other scheduling are preemptive.
- **Important CPU scheduling Terminologies**:
  - **Burst Time/Execution Time (Duration)**: It is a time required by the process to complete execution. It is also called running time.
  - **Arrival Time**: when a process enters in a ready state.
  - **Finish Time**: when process complete and exit from a system.
  - **Turn around time**: Completion time-arrival time
  - **Waiting time**: Turn around time- burst time.
  - **Multiprogramming**: A number of programs which can be present in memory at the same time.
  - **Jobs**: It is a type of program without any kind of user interaction.
  - **User**: It is a kind of program having user interaction.
  - **Process**: It is the reference that is used for both job and user.
  - **CPU/IO burst cycle**: Characterizes process execution, which alternates between CPU and I/O activity. CPU times are usually shorter than the time of I/O.

Scheduling algorithms:

- There are mainly six types of process scheduling algorithms,
  - First Come First Serve (FCFS)
  - Shortest-Job-First (SJF) Scheduling
  - Shortest Remaining Time
  - Priority Scheduling
  - Round Robin Scheduling
  - Multilevel Queue Scheduling

First Come First Serve (FCFS):

- First Come First Serve is the full form of **FCFS**. It is the easiest and most simple CPU scheduling algorithm. In this type of algorithm, the process which requests the CPU gets the CPU allocation first. This scheduling method can be managed with a **FIFO queue**.
- As the process enters the ready queue, its PCB (Process Control Block) is linked with the tail of the queue. So, when CPU becomes free, it should be assigned to the process at the beginning of the queue.
- **Criteria: Arrival time**
- **Characteristics of FCFS method**:
  - Jobs are executed on first come, first serve basis.
  - It is a **non-preemptive**, pre-emptive scheduling algorithm.
  - Easy to understand and implement.
  - Its implementation is based on FIFO queue.
  - Poor in performance as average wait time is high.

•**Arrival time (AT)** − Arrival time is the time at which the process arrives in ready queue.

•**Burst time (BT) or CPU time of the process** − Burst time is the unit of time in which a particular process completes its execution.

•**Completion time (CT)** − Completion time is the time at which the process has been terminated.

•**Turn-around time (TAT)** − The total time from arrival time to completion time is known as turn-around time. TAT can be written as,

•**Turn-around time (TAT) = Completion time (CT) – Arrival time (AT)** or, **TAT = Burst time (BT) + Waiting time (WT)**

•**Waiting time (WT)** − Waiting time is the time at which the process waits for its allocation while the previous process is in the CPU for execution. WT is written as,

Waiting time (WT) = Turn-around time (TAT) – Burst time (BT)

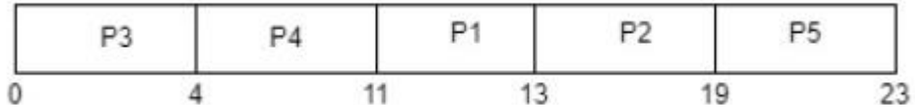
•**Response time (RT)** − Response time is the time at which CPU has been allocated to a particular process first time.

•In case of non-preemptive scheduling, generally Waiting time and Response time is same.

•**Gantt chart** − Gantt chart is a visualization which helps to scheduling and managing particular tasks in a project. It is used while solving scheduling problems, for a concept of how the processes are being allocated in different algorithms.

Cont....

- Problem 1:
- Consider the given table below and find Completion time (CT), Turn-around time (TAT), Waiting time (WT), Response time (RT), Average Turn-around time and Average Waiting time.
- Solution:
- Gantt chart

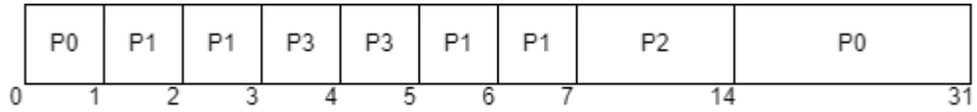


Process ID	Arrival time	Burst time	CT	TAT=CT-AT	WT=TAT-BT
P1	2	2	13	13-2= 11	11-2= 9
P2	5	6	19	19-5= 14	14-6= 8
P3	0	4	4	4-0= 4	4-4= 0
P4	0	7	11	11-0= 11	11-7= 4
P5	7	4	23	23-7= 16	16-4= 12

- Average Waiting time = (9+8+0+4+12)/5 = 33/5 = 6.6 time unit (time unit can be considered as milliseconds)
- Average Turn-around time = (11+14+4+11+16)/5 = 56/5 = 11.2 time unit (time unit can be considered as milliseconds)

Shortest Job First (SJF):

- This is also known as shortest job next, or SJN.
- This is a **non-preemptive**, pre-emptive scheduling algorithm.
- Best approach to minimize waiting time. **Criteria: Burst time**
- Easy to implement in Batch systems where required CPU time is known in advance.
- Impossible to implement in interactive systems where required CPU time is not known.
- The processer should know in advance how much time process will take.
- Given: Table of processes, and their Arrival time, Execution time



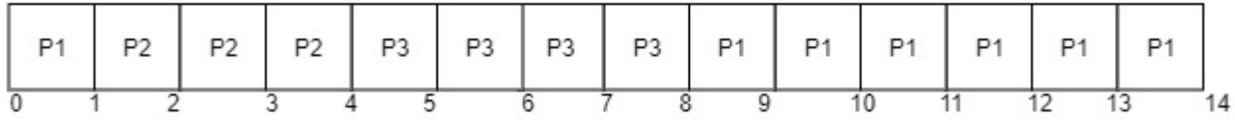
PROCESS ID	ARRIVAL TIME	BURST TIME	COMPLETION TIME	TURN AROUND TIME	WAITING TIME
P0	0	18	31	31	13
P1	1	4	5	4	0
P2	2	7	14	12	5
P3	3	2	7	4	2

Shortest Remaining Time First:

- Shortest remaining time (SRT) is the preemptive version of the SJF algorithm.
- The processor is allocated to the job closest to completion but it can be preempted by a newer ready job with shorter time to completion.
- **Criteria: Burst time**
- **Mode: Preemptive**
- Impossible to implement in interactive systems where required CPU time is not known.
- It is often used in batch environments where short jobs need to give preference.

Process	Burst Time	Arrival Time
P1	7	0
P2	3	1
P3	4	3

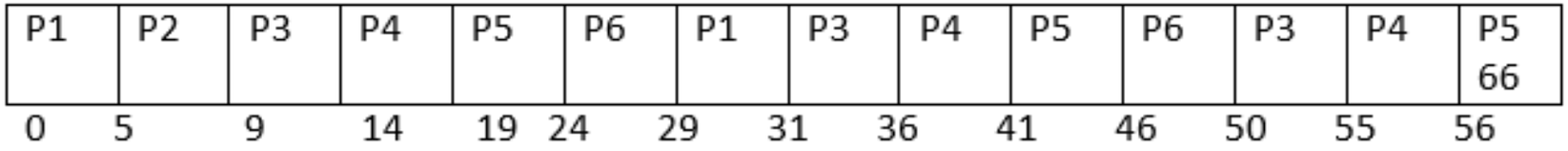
The Gantt Chart for SRTF will be:



Process	Arrival Time	Burst Time	Completion time	Turn around Time Turn Around Time = Completion Time – Arrival Time	Waiting Time Waiting Time = Turn Around Time – Burst Time
P1	0	7	14	14-0=14	14-7=7
P2	1	3	4	4-1=3	3-3=0
P3	3	4	8	8-3=5	5-4=1

Round Robin Scheduling:

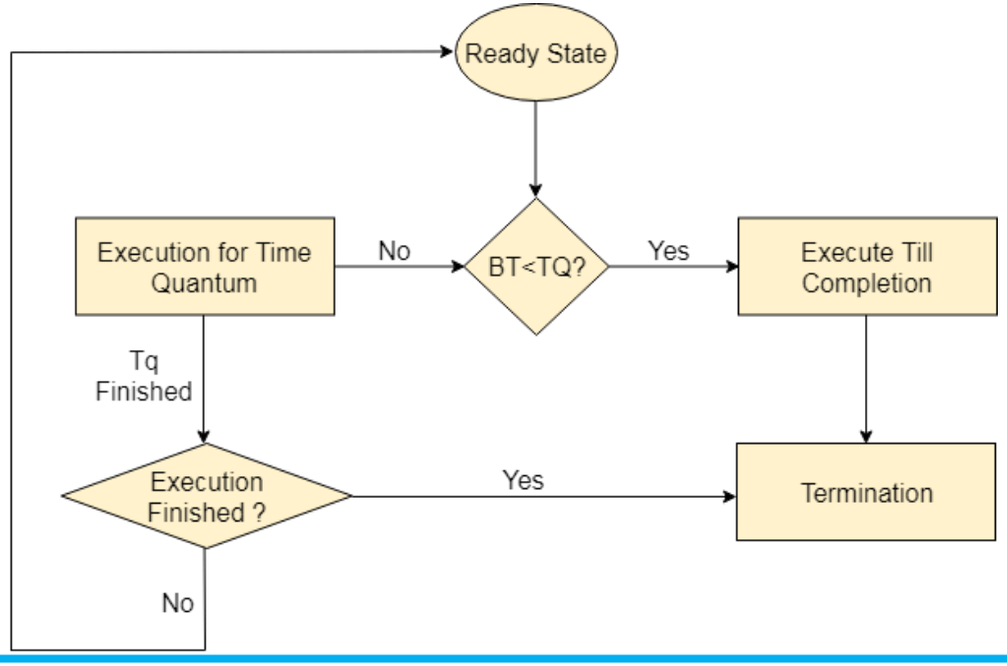
- Round Robin is the **preemptive** process scheduling algorithm.
- Each process is provided a fix time to execute, it is called a **time-quantum**.
- Once a process is executed for a given time period, it is preempted and other process executes for a given time period.
- **Context switching** is used to save states of preempted processes.
- Assume Time Quantum TQ = 5
- Ready Queue: P1, P2, P3, P4, P5, P6, P1, P3, P4, P5, P6, P3, P4, P5



Process ID	Arrival Time	Burst Time	Completion Time	Turn Around Time	Waiting Time
P1	0	7	31	31	24
P2	1	4	9	8	4
P3	2	15	55	53	38
P4	3	11	56	53	42
P5	4	20	66	62	42
P6	4	9	50	46	37

Cont....

- First, the processes which are eligible to enter the ready queue enter the ready queue. After entering the first process in Ready Queue is executed for a Time Quantum chunk of time. After execution is complete, the process is removed from the ready queue. Even now the process requires some time to complete its execution, then the process is added to Ready Queue.
- The Ready Queue does not hold processes which already present in the Ready Queue. The Ready Queue is designed in such a manner that it does not hold non unique processes. By holding same processes Redundancy of the processes increases.
- After, the process execution is complete, the Ready Queue does not take the completed process for holding.

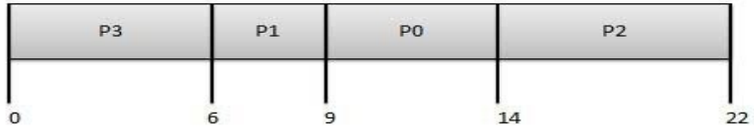


Priority Based Scheduling:

- Priority scheduling is a non-preemptive algorithm and one of the most common scheduling algorithms in batch systems. In some cases it is preemptive.
- Each process is assigned a priority. Process with highest priority is to be executed first and so on. **Criteria**: **priority**
- Processes with same priority are executed on first come first served basis.
- Priority can be decided based on memory requirements, time requirements or any other resource requirement.
- Given: Table of processes, and their Arrival time, Execution time, and priority. Here we are considering 1 is the lowest priority.

Process	Arrival Time	Execution Time	Priority	Service Time
P0	0	5	1	0
P1	1	3	2	11
P2	2	8	1	14
P3	3	6	3	5

Process	Arrival Time	Execute Time	Priority	Service Time
P0	0	5	1	9
P1	1	3	2	6
P2	2	8	1	14
P3	3	6	3	0



Waiting time of each process is as follows –

Process	Waiting Time
P0	0 - 0 = 0
P1	11 - 1 = 10
P2	14 - 2 = 12
P3	5 - 3 = 2

Average Wait Time: (0 + 10 + 12 + 2)/4 = 24 / 4 = 6

Multiple-Level Queues Scheduling:

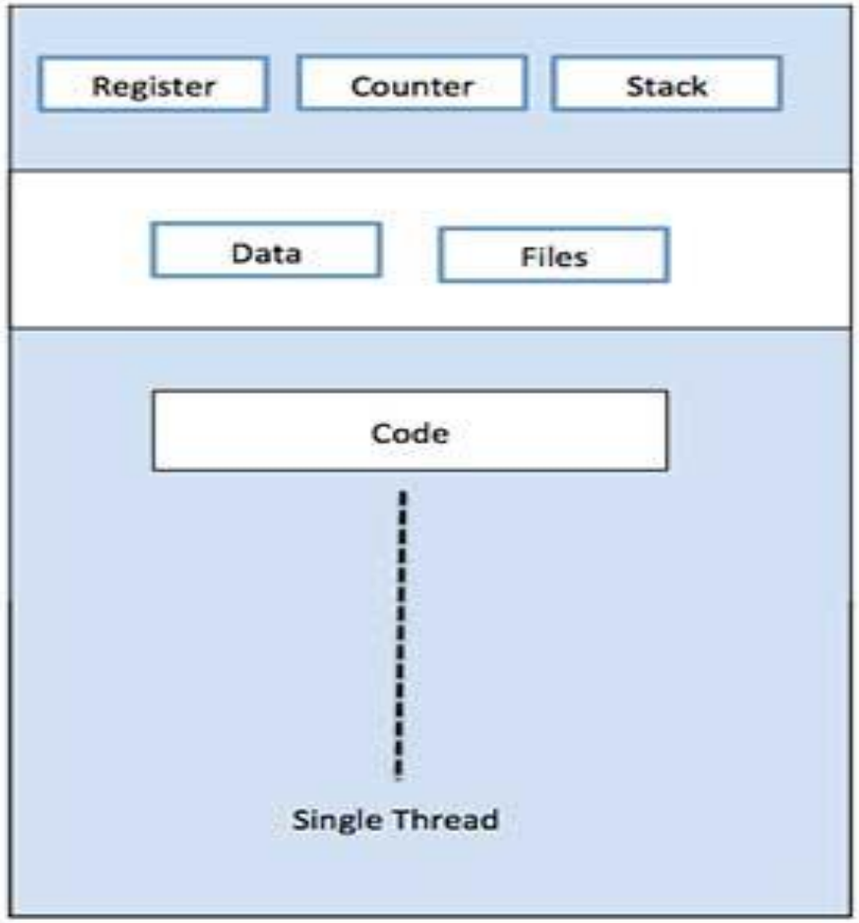
- Multiple-level queues are not an independent scheduling algorithm. They make use of other existing algorithms to group and schedule jobs with common characteristics.
  - Multiple queues are maintained for processes with common characteristics.
  - Each queue can have its own scheduling algorithms.
  - Priorities are assigned to each queue.
- For example, CPU-bound jobs can be scheduled in one queue and all I/O-bound jobs in another queue. The Process Scheduler then alternately selects jobs from each queue and assigns them to the CPU based on the algorithm assigned to the queue.

Operating system services for process and thread management:

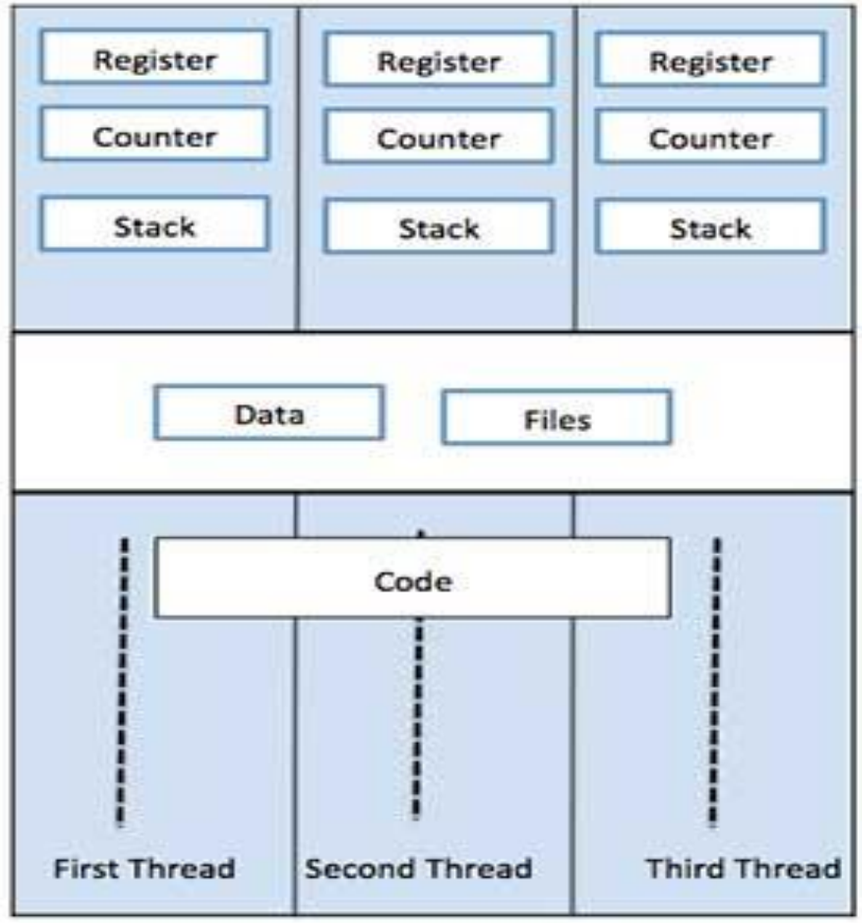
- **Threads**:
  - A thread is **a flow of execution through the process code**, **with its own program counter** that keeps track of which instruction to execute next, system registers which hold its current working variables, and a stack which contains the execution history. It comprises **a thread ID, program counter, register set, and stack**.
  - A thread shares with its peer threads few information like **code segment, data segment and open files**.
  - When one thread alters a code segment memory item, all other threads see that.
  - A thread is also called a **lightweight process**. Threads provide a way to improve application performance through parallelism.
  - Threads represent a software approach for improving performance of operating system by reducing the overhead thread is equivalent to a classical process.
  - Each thread belongs to exactly one process and no thread can exist outside a process. Each thread represents a separate flow of control.
  - Threads have been successfully used in implementing network servers and web server.
  - They also provide a suitable foundation for parallel execution of applications on shared memory multiprocessors.

Cont.…

- The following figure shows the working of a single-threaded and a multithreaded process.



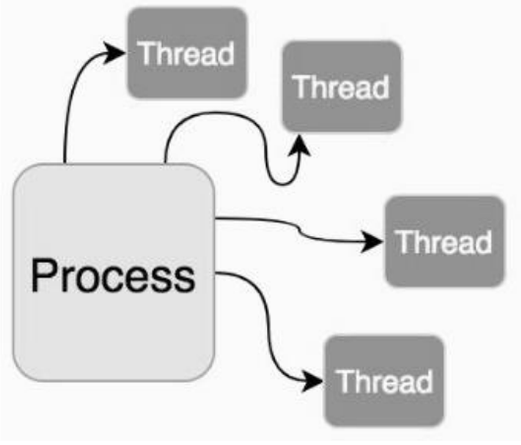
Single Process P with single thread



Single Process P with three threads

Cont.…

- Multithreading is a process execution model in which the process is dispatched in multiple lightweight processes called (threads).
- Each thread can be executed independently, and it can share some or all the resources of a process.
- Depending on the system hardware, threads can run in parallel on multicore or multiprocessor systems and can run concurrently on a single processor by using the time-sharing mechanisms.
- Multithreading enriches operating systems by improving their performance and execution capabilities.
- Multithreading models are three types
  - Many to many relationship.
  - Many to one relationship.
  - One to one relationship.



Cont.…

- In operating systems, a process is the unit of resource ownership and protection, and a thread is a unit of dispatching.
- As an independent lightweight process, threads have accounting and control data related to them.
- Operating systems maintain Thread Control Block (TCB) for each thread. Also, each thread is assigned a user stack and a kernel stack space.
- Depending on the type of threads, the life cycle of a thread goes through a number of states.
- Types of threads: Multithread designs include two types of threads in operating systems: user-level and kernel-level threads.
  - User Level Threads (ULTs) are part of the user application design and are implemented using multithreading libraries.
  - Kernel-level threads (KLTs) are lightweight units of kernel scheduling in which one or more threads exist within each process. In most contemporary systems, both types of threads exist separately at the same time. Therefore, a mapping between the ULTs and KLTs is applied differently in different operating systems.

Cont.…

- **User Level Threads**:
- User Level Thread is a type of thread that is not created using system calls. The kernel has no work in the management of user-level threads. User-level threads can be easily implemented by the user. In case when user-level threads are single-handed processes, kernel-level thread manages them. Let's look at the advantages and disadvantages of User-Level Thread.
- Advantages of User-Level Threads
  - Implementation of the User-Level Thread is easier than Kernel Level Thread.
  - Context Switch Time is less in User Level Thread.
  - User-Level Thread is more efficient than Kernel-Level Thread.
  - Because of the presence of only Program Counter, Register Set, and Stack Space, it has a simple representation.
- Disadvantages of User-Level Threads
  - There is a lack of coordination between Thread and Kernel.
  - In case of a page fault, the whole process can be blocked.

Cont….

- **Kernel Level Threads:**
- A kernel Level Thread is a type of thread that can recognize the Operating system easily. Kernel Level Threads has its own thread table where it keeps track of the system. The operating System Kernel helps in managing threads. Kernel Threads have somehow longer context switching time. Kernel helps in the management of threads.

- Advantages of Kernel-Level Threads
  - It has up-to-date information on all threads.
  - Applications that block frequency are to be handled by the Kernel-Level Threads.
  - Whenever any process requires more time to process, Kernel-Level Thread provides more time to it.
- Disadvantages of Kernel-Level threads
  - Kernel-Level Thread is slower than User-Level Thread.
  - Implementation of this type of thread is a little more complex than a user-level thread.

Cont.…

- **Benefits of Thread in Operating System:**
  - **Responsiveness**: If the process is divided into multiple threads, if one thread completes its execution, then its output can be immediately returned.
  - **Faster context switch**: Context switch time between threads is lower compared to the process context switch. Process context switching requires more overhead from the CPU.
  - **Effective utilization of multiprocessor system**: If we have multiple threads in a single process, then we can schedule multiple threads on multiple processors. This will make process execution faster.
  - **Resource sharing**: Resources like code, data, and files can be shared among all threads within a process. Note: Stacks and registers can’t be shared among the threads. Each thread has its own stack and registers.
  - **Communication**: Communication between multiple threads is easier, as the threads share a common address space. while in the process we have to follow some specific communication techniques for communication between the two processes.
  - **Enhanced throughput of the system**: If a process is divided into multiple threads, and each thread function is considered as one job, then the number of jobs completed per unit of time is increased, thus increasing the throughput of the system.

Cont....

- **Difference between Process and Thread:**

S.N.	Process	Thread
1	Process is heavy weight or resource intensive.	Thread is light weight, taking lesser resources than a process.
2	Process switching needs interaction with operating system.	Thread switching does not need to interact with operating system.
3	In multiple processing environments, each process executes the same code but has its own memory and file resources.	All threads can share same set of open files, child processes.
4	If one process is blocked, then no other process can execute until the first process is unblocked.	While one thread is blocked and waiting, a second thread in the same task can run.
5	Multiple processes without using threads use more resources.	Multiple threaded processes use fewer resources.
6	In multiple processes each process operates independently of the others.	One thread can read, write or change another thread's data.

Thank You !!!